



**MULTI-OBSERVATION CONTINUOUS DENSITY HIDDEN MARKOV MODELS  
FOR ANOMALY DETECTION IN FULL MOTION VIDEO**

THESIS

Matthew P. Ross, Major, USAF

AFIT/GCS/ENG/12-07

**DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY**

***AIR FORCE INSTITUTE OF TECHNOLOGY***

**Wright-Patterson Air Force Base, Ohio**

DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE;  
DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT/GCS/ENG/12-07

MULTI-OBSERVATION CONTINUOUS DENSITY HIDDEN MARKOV MODELS  
FOR ANOMALY DETECTION IN FULL MOTION VIDEO

THESIS

Presented to the Faculty  
Department of Electrical and Computer Engineering  
Graduate School of Engineering and Management  
Air Force Institute of Technology  
Air University  
Air Education and Training Command  
in Partial Fulfillment of the Requirements for the  
Degree of Master of Science

Matthew P. Ross, B.S.C.S.

Major, USAF

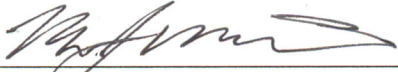
June 2012

DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE;  
DISTRIBUTION UNLIMITED


MULTI-OBSERVATION CONTINUOUS DENSITY HIDDEN MARKOV MODELS  
FOR ANOMALY DETECTION IN FULL MOTION VIDEO

Matthew P. Ross, B.S.C.S.  
Major, USAF


Approved:

  
Lt Col Brett J. Borghetti, PhD (Co-Chairman)


29 MAY 2012  
Date

  
Angela A. Sodemann, PhD (Co-Chairman)

29 May 2012  
Date

  
Jeffery K. Cochran, PhD (Committee Member)

29 May 12  
Date

  
Lt Col Jeffrey D. Clark, PhD (Committee Member)

29 May 12  
Date

**Abstract**

An increase in sensors on the battlefield produces an abundance of collected data that overwhelms the processing capability of the DoD. Automated Visual Surveillance (AVS) seeks to use machines to better exploit increased sensor data, such as by highlighting anomalies. In this thesis, we apply AVS to overhead Full Motion Video (FMV). We seek to automate the classification of soldiers in a simulated combat scenario into their agent types. To this end, we use Multi-Dimensional Continuous Density Hidden Markov Models (MOCDHMMs), a form of HMM which models a training dataset more precisely than simple HMMs. MOCDHMMs are theoretically developed but thinly applied in literature. We discover and correct three errors which occur in HMM algorithms when applied to MOCDHMMs but not when applied to simple HMMs. We offer three fixes to the errors and show analytically why they work. To show the fixes effective, we conduct experiments on three datasets: two pilot experiment datasets and a simulated combat scenario dataset. The modified MOCDHMM algorithm gives statistically significant improvement over the standard MOCDHMM: 5% improvement in accuracy for the pilot datasets and 3% for the combat scenario dataset. In addition, results suggest that increasing the number of hidden states in an MOCDHMM classifier increases the separability of the classes but also increases classifier bias. Furthermore, we find that classification based on tracked position alone is possible and that MOCDHMM classifiers are highly resistant to noise in their training data.

*For my wife. I didn't always know we'd get through this, but you knew we would. Thanks.*

## Table of Contents

	Page
Abstract . . . . .	iv
Dedication . . . . .	v
Table of Contents . . . . .	vi
List of Figures . . . . .	ix
List of Tables . . . . .	x
List of Symbols . . . . .	xi
 I. Introduction and Motivation . . . . .	 1
1.1 Strategic Implications of Full Motion Video . . . . .	1
1.2 Limits of FMV: Too much data and too few analysts . . . . .	3
1.3 Reducing human workload: anomaly detection . . . . .	5
1.4 Scope of effort . . . . .	7
1.5 Contributions . . . . .	8
 II. Background and Related Work . . . . .	 10
2.1 Related Work in Automated Surveillance . . . . .	10
2.1.1 Terminology . . . . .	11
2.1.2 State of the Art . . . . .	12
2.1.3 Object Based Decomposition Methods . . . . .	14
2.1.3.1 Experiments Involving Object Based Decomposition . . . . .	14
2.2 Related work in Discrete or Single-Variable Hidden Markov Models . . . . .	17
2.2.1 Related works which reduce to 1-D observations . . . . .	17
2.2.2 Related works which discretize observations . . . . .	18
2.2.3 Related works which reduce to 1-D observations AND discretize observations . . . . .	 18
2.3 Multi Observation Continuous Density Hidden Markov Models . . . . .	19
2.3.1 Why MOCDHMMs are a good choice in Automated Visual Surveillance . . . . .	 20
2.3.2 Related work in MOCDHMMs . . . . .	20
2.4 Automated Video Surveillance: Conclusion . . . . .	22

	Page
III. Methodology . . . . .	23
3.1 Preprocessing entity tracks in Full Motion Video (FMV) . . . . .	23
3.2 Classification by HMM . . . . .	24
3.3 Standard MOCDHMM Algorithm . . . . .	25
3.3.1 Hidden Markov Models and the Markovian Assumption . . . . .	26
3.3.2 Generalizing the HMM: Multi-Observation and Continuous Den- sity Observations . . . . .	27
3.3.3 Segmental K Means Training . . . . .	28
3.3.4 Forward Backward algorithm . . . . .	29
3.4 Mathematical problems with standard MOCDHMM algorithm . . . . .	33
3.4.1 Error 1: Badly conditioned covariance matrices . . . . .	33
3.4.2 Error 2: Division by zero in alpha scaling . . . . .	34
3.4.3 Error 3: Biased classifier . . . . .	35
3.5 Modified MOCDHMM algorithm . . . . .	37
3.5.1 Proposed fix for error 1: Moore Penrose Pseudoinverse in Multivariate Gaussian PDF . . . . .	37
3.5.2 Proposed fix for error 2: Return negative infinity if the scaling factor is zero . . . . .	39
3.5.3 Proposed fix for error 3: Use scored evaluation instead of raw log- likelihood . . . . .	40
3.6 Experimental Design . . . . .	41
3.6.1 Unblended Pilot Experimental Design . . . . .	42
3.6.2 Blended Pilot Experimental Design . . . . .	45
3.6.3 Fryer Dataset Experimental Design . . . . .	47
3.6.3.1 Preprocessing the Fryer dataset . . . . .	49
IV. Results and Discussion . . . . .	51
4.1 Experimental examples of errors arising from standard MOCDHMM algorithm . . . . .	51
4.1.1 Example of error 1: badly scaled covariance matrix . . . . .	51
4.1.2 Example of error 2: division by zero in alpha scaling . . . . .	52
4.1.3 Example of error 3: biased classifier . . . . .	55
4.2 Results of experiment with standard MOCDHMM algorithm . . . . .	56
4.2.1 Unblended pilot experiment results: Standard MOCDHMM . . . . .	56
4.2.2 Blended pilot experiment results: Standard MOCDHMM . . . . .	56
4.2.2.1 Performance of blended pilot experiment: Standard MOCDHMM . . . . .	57
4.2.2.2 Characterizing the error resistance of MOCDHMMs . . . . .	60
4.2.3 Fryer dataset experiment results: standard MOCDHMM . . . . .	61
4.3 Results of experiments with modified MOCDHMM algorithm . . . . .	62



	Page
4.3.1 Unblended pilot experiment results: Modified MOCDHMM . . . . .	63
4.3.2 Blended pilot experiment results: Modified MOCDHMM . . . . .	64
4.3.3 Fryer dataset experiment results: modified MOCDHMM . . . . .	66
4.3.3.1 Remarks on statistical significance of modified MOCDHMM Fryer dataset results . . . . .	72
4.3.3.2 Comparison of modified MOCDHMM algorithm to Mean Euclidean Distance (MED) classification for the Fryer dataset . . . . .	73
V. Conclusions and Future Work . . . . .	76
5.1 Summary of findings . . . . .	76
5.1.1 Analytical contributions . . . . .	76
5.1.2 Empirical findings . . . . .	77
5.2 Advantages and limitations of MOCDHMMs . . . . .	78
5.3 Future work . . . . .	80
5.3.1 Improve countermeasures for biased classifiers . . . . .	81
5.3.2 Discovering other failure modes of MOCDHMMs . . . . .	81
5.3.3 Dataset selection . . . . .	82
5.3.4 Improved distribution analysis for scoring log-likelihood . . . . .	83
5.3.5 Automatic model selection based on AIC or BIC . . . . .	85
5.3.6 Implement required mathematical changes in the Baum Welch Re- estimation procedure . . . . .	86
5.3.7 Generalize experiments to permit five-at-a-time classification . . . . .	86
Appendix A: Permission to Publish Proprietary Information . . . . .	87
Appendix B: Proofs . . . . .	90
Appendix C: Background on Automated Surveillance . . . . .	92
Bibliography . . . . .	101

## List of Figures

Figure	Page
1.1 AVS procedures to produce classified entities from FMV . . . . .	8
2.1 CPOL family of solutions . . . . .	11
2.2 Timeline of innovation relevant to AVS . . . . .	13
3.1 Definitions of observation, sequence, and sequence set . . . . .	24
3.2 Depiction of a biased classifier . . . . .	36
3.3 Sequence mixed datasets used in Blended Pilot Experiment . . . . .	46
3.4 Error response profiles . . . . .	47
3.5 Method for measuring angular movement versus average direction of movement	49
3.6 Method for calculating Angular Deviation, $\Theta$ . . . . .	50
4.1 HMM produced by K Means Learning for agent H . . . . .	54
4.2 Results of the blended pilot experiment: standard algorithm . . . . .	58
4.3 Results of the blended pilot experiment: modified algorithm . . . . .	65
5.1 Histogram of distribution of log-likelihoods $\mathcal{L}(\mathcal{S}^{\text{AWARE}} \lambda_{\text{AWARE}})$ . . . . .	84
C.1 Sample ROC Curve . . . . .	98

## List of Tables

Table	Page
1.1 Phases of Intelligence Exploitation . . . . .	6
3.1 GMM Definitions for pilot experiment agents . . . . .	43
3.2 Names of the nine simulated agents for the pilot experiments . . . . .	43
4.1 Unscaled forward variable matrix for calculation of $\alpha_t(i)$ . . . . .	53
4.2 Results of unblended pilot experiment for standard MOCDHMM: EWA accuracy	57
4.3 Results summary for blended pilot experiment: standard algorithm . . . . .	61
4.4 Accuracies for Fryer experiment: two-state standard algorithm . . . . .	62
4.5 Results of unblended pilot experiment for modified MOCDHMM: EWA accuracy	63
4.6 Results summary for blended pilot experiment: modified algorithm . . . . .	66
4.7 Accuracy summary for all Fryer experiments . . . . .	67
4.8 Accuracies for Fryer experiment: five-state modified-unscored . . . . .	68
4.9 Mean log-likelihood matrix for two hidden states . . . . .	70
4.10 Mean log-likelihood matrix for five hidden states . . . . .	70
4.11 Overall improvement of modified MOCDHMM algorithm . . . . .	74

## List of Symbols

- $\mathbf{O}_t$  Observation vector at timestep  $t$
- $\mathcal{S} = (\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_T)$  An observation sequence
- $\mathcal{E}^G = \{\mathcal{S}_1^G, \mathcal{S}_2^G, \dots, \mathcal{S}_L^G\}$  Sequence set  $G$
- $L$  The total number sequences in a sequence set
- $s = (s_1, s_2, \dots, s_T)$  A hidden state sequence that generates  $\mathcal{S}$  in an HMM
- $s_t$  The hidden state at time  $t$
- $\mathcal{S}_1^G$  Sequence 1 of sequence set  $G$
- $i$  A hidden state
- $j$  A hidden state, used as a dummy variable or variable of summation
- $N$  The total number of states in an HMM
- $t$  A timestep of a sequence
- $T$  The total number of timesteps in an observation sequence
- $\tau$  A timestep of a sequence, used as a dummy variable or variable of summation
- $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$  A Hidden Markov Model
- $a_{ij}$  Probability of being in state  $j$  at time  $t + 1$  given the process was in  $i$  at  $t$
- $b_i$  PDF for hidden state  $i$

- $B = \{b_1, b_2, \dots, b_N\}$  Set of PDFs for an HMM.
- $b_i(\mathbf{O}_t)$  Likelihood of emitting  $\mathbf{O}_t$  at hidden state  $i$
- $\pi = \{\pi_1, \pi_2, \dots, \pi_N\}$  Set of starting probabilities for an HMM
- $\pi_i$  Probability of beginning an HMM process in hidden state  $i$
- $\alpha_t(i)$  Forward variable at hidden state  $i$  for time  $t$
- $\hat{\alpha}_t(i)$  Scaled forward variable at hidden state  $i$  for time  $t$
- $\omega_t(i)$  Intermediate forward variable at hidden state  $i$  for time  $t$
- $c_t$  Scaling factor at time  $t$
- $C_t$  The product of all scaling factors  $c_\tau$  for  $\tau \in \{1, 2, \dots, t\}$
- $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix}$  Transition matrix of an HMM
- $f(\mathcal{S}|\lambda)$  Likelihood of hmm  $\lambda$  emitting sequence  $\mathcal{S}$
- $\mathcal{L}(\mathcal{S}|\lambda)$  Log-likelihood of HMM  $\lambda$  emitting sequence  $\mathcal{S}$
- $z(\mathcal{S}|\hat{\lambda})$  Scored evaluation of sequence  $\mathcal{S}$  against scored HMM  $\hat{\lambda}$
- $\hat{\lambda} = (\mathbf{A}, B, \pi, \hat{\mu}, \hat{\sigma})$  A Scored Hidden Markov Model
- $\mathcal{L}_\mu(\mathcal{E}|\lambda)$  The average log-likelihood for all sequences  $\mathcal{S} \in \mathcal{E}$  of HMM  $\lambda$  emitting  $\mathcal{S}$
- $\Lambda_{YZ}$  Classifier composed of HMMs  $Y$  and  $Z$

- $\Theta$ : Angular Deviation. A random variable, the difference in heading (in degrees) from the overall direction of movement over the sequence
- $S$ : Speed. A random variable, the speed of the agent at a given time step

# MULTI-OBSERVATION CONTINUOUS DENSITY HIDDEN MARKOV MODELS FOR ANOMALY DETECTION IN FULL MOTION VIDEO

## **I. Introduction and Motivation**

Automated Visual Surveillance (AVS) and Full Motion Video (FMV) have the potential to increase the effectiveness of human intelligence analysts. This chapter surveys research in surveillance and intelligence gathering reporting that there is too much data for complete human analysis [5–8, 29, 49]. The amount of intelligence data available grows with technological advancement in sensors and intelligence platforms. Thus, the amount of human resources required to analyze all such data is prohibitively high.

This chapter first discusses the strategic implications of FMV on the battlefield. The chapter then details the problem FMV creates—that there is too much for detailed human analysis. Following that, the chapter discusses anomaly detection through AVS as one means of reducing the burden on human analysts; we describe how the amount of video humans need to watch can be reduced to only that video which is anomalous. Finally, the chapter gives a brief overview of the mathematical models we will refine in this thesis to facilitate AVS.

### **1.1 Strategic Implications of Full Motion Video**

The prevalence of overhead FMV on the battlefield can provide advantages for the U.S. and its allies if used properly. As an intelligence medium, FMV has the potential to impact: 1) mission planning [5]; 2) real-time targeting, tasking, and retasking [5]; 3) commander sensitivity to cultural, social, economic, and political interconnections [5]; and 4) force protection and counter-terrorism [29].

First, FMV positively impacts mission planning. Mission planners use FMV to focus assets on geographic points of greatest interest. For example, if intelligence ties adversary activity to a particular building, then observed movement from that building to another may demonstrate adversary activity at the second building [5]. Such movement may indicate need for force application (or more intelligence collection) around such particular points and minimize use of assets where no adversary activity is indicated. Moreover, mission planners use FMV to minimize civilian casualties. For example, if intelligence gleaned from video data shows that insurgents regularly travel to a point away from civilian activity, such insurgents can be engaged where and when collateral damage is unlikely [5].

Second, FMV enables rapid discovery of hostile activity among both insurgent and large-scale military adversaries. Wide area video can depict large-scale movement such as fleeing civilians or civil unrest. When these occur, command and control elements can be notified and appropriate action taken. Furthermore, FMV can uncover specific small-scale events such as the placement of an improvised explosive device (IED) [5]. In addition, full motion video taken during operations can alert operators and commanders of changing conditions. For example, FMV analysis might reveal cases in which insurgents depart combat emplacements and enter vehicles in an attempt to flee. This condition, when detected, could aid in real-time retasking of ground and air assets.

Third, full motion video can reveal movement among important economic centers, religious sites, and residences. Analysis of such data may reveal social, economic, political, and cultural interconnections among a population. A deeper understanding of these interconnections promotes cultural sensitivity among friendly forces. Such sensitivity produces desirable battlefield effects while at the same time avoiding actions that may lead to embarrassment [5]. For example, a campaign may have an objective to convince a population that the U.S. action is necessary and moral. However, if efforts to convince a particular population are combined with bombings of places they find important then the



effort will be counterproductive. Automated analysis of full motion video may reveal these cultural interconnections.

Fourth, FMV aids in force protection and counter-terrorism, domains in which military security has overlapping interests with civilian security. In force protection and counter-terrorism, security seeks to monitor activities around a place to defend it. FMV can reveal suspicious actions so that security may be dispatched to investigate. FMV is also used to analyze past actions to determine when an illegal or dangerous act (such as Improvised Explosive Device (IED) placement) occurred. This information may help identify who placed the IED or where they went after having done so. FMV use to analyze the past is called *video forensics* [29].

## **1.2 Limits of FMV: Too much data and too few analysts**

Civilian law enforcement and military application of force are two domains that use FMV. They face a common set of problems: overwhelming amount of data, shortage of available analysts, and limitations related to human factors.

In civilian applications, video surveillance has seen widespread use; in certain countries (like the U.K.) it has become ubiquitous. The U.K. now has one camera for every 17 citizens. To date, most video surveillance systems are used by private organizations like corporations and private security firms. However, public law enforcement agencies install FMV cameras increasingly as legal restrictions are relaxed [6].

Civilian use of video surveillance systems has allowed increased security and better forensic analysis of criminal activity, but its effectiveness is limited because it is human-labor intensive. U.K. estimates that only 1/4 to 1/78 of all video data is watched even once by human operatives [7].

Research into civilian FMV-based security reveals human factors limitations: attention-intensiveness and boredom. Most video data contains no information of interest [6]. A U.K. Police Scientific Development Branch study claims that operatives in general cannot

effectively monitor more than four screens simultaneously [49]. The ratio of operatives to screens in the U.K. is 1:16 [7]. Furthermore, a lack of centralized control in civilian applications often leaves operatives not knowing what they are looking for. This opens FMV based security to further vulnerabilities. Illegal profiling and missing items of interest due to *cognitive bias* are all real possibilities. Cognitive bias is a pattern of judgment (in this case, on the part of security operatives) that leads to inaccurate perception in specific situations [27]. For example, a belief that people dressed a certain way are more likely to be troublemakers would be an example of cognitive bias if, in fact, people dressed that way were no more likely to cause trouble. Clearly, the realities of video surveillance in civilian application demand research into computer automated systems.

The challenges to FMV use in military applications are parallel to those in civilian applications. Kuperman [8], in his overview of the Distributed Common Ground System (DCGS), details limitations affecting military capability. For typical U-2 spy plane application, 45-47 people are required at all times to analyze incoming data. Typically, planners provide 24 hour analyst coverage by establishing two shifts of 12 hours each, thus demanding almost 100 trained personnel. Thus, not including short breaks, personnel routinely spend 12 hours looking at computer screens with no pause or rewind capability. Personnel who perform this function have summed it up shortly: “It’s painful” [8].

The large amount of incoming data hampers the application of two important concepts in intelligence doctrine: Intelligent Queuing and Phased Exploitation [24]. First, by “Intelligent Queuing” intelligence personnel ensure important and time critical data is analyzed first. Second, by “phased exploitation” personnel ensure that small, important portions of video data are analyzed more deeply; doctrine divides exploitation into phases one through three according to Table 1.1. Intelligence analysts use the table as a framework to prioritize work.

Table 1.1 contains three levels of intelligence exploitation. *Tactical/real-time* exploitation refers to intelligence which will aid only in a particular, short-term skirmish or battle [24]. For example, tactical/real-time exploitation might produce information that insurgents have fled to a particular point; such information is of interest only until the insurgents move again. *Operational exploitation* refers to intelligence that may inform battlefield decision making over the course of a long-term conflict [24]. For example, operational exploitation might produce information that insurgents typically flee to certain types of buildings when threatened; such information is of interest until insurgents make changes to their tactics. *Strategic exploitation* refers to intelligence that may inform the methods commanders use to impact the cultural or political environment [24]. For example, strategic exploitation might produce information that fighting in urban areas increases a civilian population's hostility to U.S. troops so much that it is not worth pursuing insurgents into buildings; such information would be useful throughout a conflict since cultural norms change very slowly. (*Scientific and Technical Exploitation* is described in the Joint Publication 2-0 concerning intelligence [24] but not relevant to this thesis.)

Phases two and three (operational and strategic exploitation) involve higher commitments of time and resources. Therefore, data receives phase two or phase three exploitation only if it will likely yield 1) highly-valuable intelligence or 2) intelligence which is useful for a long period of time. However, little FMV data is ever analyzed outside of phase one because there is so much FMV data and it is difficult to determine what data bears further exploitation [8]. The human-labor intensive process implies that as personnel budgets shrink, computer automation is desirable [24].

### **1.3 Reducing human workload: anomaly detection**

Human analysis of all FMV is prohibitively expensive and error-prone. A body of machine learning research seeks to automate some part of the FMV analysis process,

Table 1.1: Phases of Intelligence Exploitation

Phase	Time Period	Description
1	Hours-Days	Tactical/Real-time Exploitation
2	Days-Weeks	Operational Exploitation
3	Months-Years	Strategic or Scientific and Technical Exploitation

thereby reducing the amount of human effort involved. One technique used in such an automation is *anomaly detection*. Anomaly detection is a form of AVS which separates the common from the uncommon in intelligence data. This section cites studies to suggest that anomalous behavior is behavior of interest in FMV. These studies claim that one vital step in reducing the amount of video needing to be watched is to eliminate everything normal and highlight that which is anomalous.

Norris and Armstrong [36] produce a comprehensive report on surveillance installations from their personal study of two public installations in the U.K. According to their 7 working rules of video surveillance, trouble arises most typically where people or things are *out of place* or *out of time*.

Military force protection literature echoes the *out of place* or *out of time* claim. The U.S. Air Force Eagle Eyes program (which provides reporting mechanisms for suspicious or criminal behavior) asks members and dependents to report people out of place. This guidance from an Air Force security agency lends support to the claim that users of automated FMV systems desire to cull out the anomalous sequences from FMV. Eagle Eyes [1] defines *out of place* on its website thus:

People who don't seem to belong in the workplace, neighborhood, business, establishment, or anywhere else. It includes suspicious border crossings and stowaways aboard ships or people jumping ship in port. This category [persons out of place] is hard to define, but the point is that people know what looks right

and what doesn't look right in their neighborhoods, office spaces, commutes, etc., and if a person just doesn't seem like he or she belongs, there's probably a reason for that.

Note that we make no attempt in this thesis to determine whether anomalies are *in reality* more likely to be interesting acts. Instead, we make the assumption that users of FMV wish to identify anomalous sequences within larger sets of video data. Thus, we aim to provide a mathematically rigorous way to automate the anomaly detection process.

#### **1.4 Scope of effort**

The research of this thesis seeks to expand the tools available to perform anomaly detection in FMV. Pairwise classification based on tracks is the specific area in which most work is performed. The work described here assumes that tracks are already obtained from full motion video; we do not attempt to advance the tracking process.

This research does not seek to classify tracks into *normal* and *anomalous*. However, the work performed is still relevant to the field of anomaly detection. Some anomaly detection methods take the approach of learning what is normal and alerting when something abnormal occurs; others classify various entities into the classes *normal* and *anomalous*. This thesis provides groundwork to support the latter method by classifying experimental datasets designed to emulate the behavior of individuals in an FMV dataset.

Specifically, this research modifies a specific mathematical structure to improve classification performance. The chain of procedures to produce classified tracks from raw FMV video data is given in Figure 1.1, where a rhombus represents a dataset and a rectangle represents a procedure. We seek to improve the process of classification to produce classified tracks.

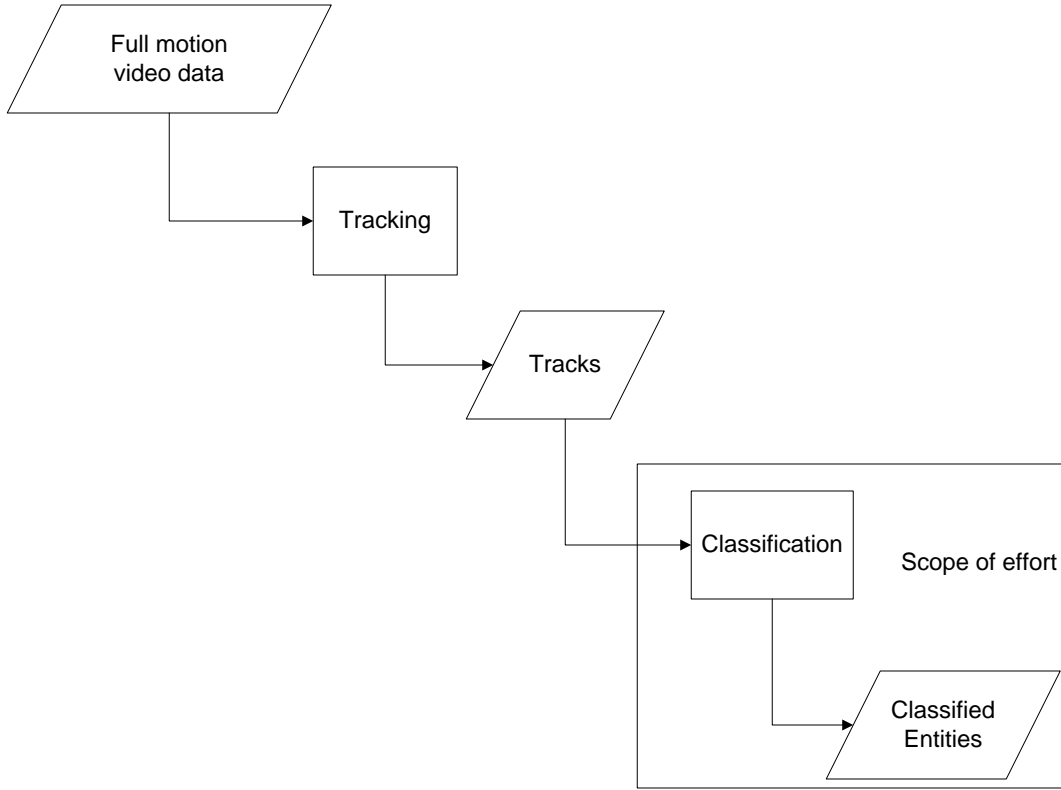


Figure 1.1: Summary of AVS procedures to produce classified entities from FMV. The scope of the work in this thesis is annotated.

## 1.5 Contributions

Previous work into automated anomaly detection has focused primarily on methods that discretize high dimensional continuous spaces into discrete symbols. Discretization (of which quantization is one form) reduces the amount of information to be processed by a modeling technique. However, the discretization process also results in ignoring information which may potentially provide class separation. In machine learning, *class separation* is a numerical difference in the features of two objects to be classified. Class separation is a key to reliable automated classification.

The research of this thesis relaxes the requirement that incoming data be discrete symbols in order to make discretization unnecessary and preserve the information discarded

by discretization. Specifically, this research uses a Hidden Markov Model (HMM) which emits vectors from multi-dimensional continuous vector spaces. We term the resulting structure a *Multi-Observation Continuous Density Hidden Markov Model* (MOCDHMM).

Current literature treats the MOCDHMMs thinly, instead favoring techniques which use vector quantization and discrete-observation models. This may be due to the mathematical complexities exposed when considering  $n$ -dimensional continuous vectors instead of one-dimensional discrete symbols as observations. Several such complexities are encountered and detailed here, including near-singular covariance matrices in training, division by zero errors in HMM algorithms generalized for continuous spaces, and bias in HMM-based classifiers. We propose several solutions to the encountered problems.

The work performed in this thesis makes MOCDHMMs a more viable option in anomaly detection, removing limitations that require discretization and possible information loss.

## **II. Background and Related Work**

This chapter presents an overview of the current state of the art in Automated Visual Surveillance (AVS) and Hidden Markov Models (HMMs). Section 2.1 discusses papers representing the state of AVS research as a whole. Section 2.2 describes advances using HMMs other than MOCDHMMs. Section 2.3 treats a few studies into MOCDHMMs.

Several papers are cited here to suggest that most applications of HMMs use either single variable models or discretized observations. Multi-Observation Continuous Density HMMs (MOCDHMMs) have received little attention.

### **2.1 Related Work in Automated Surveillance**

The Air Force Institute of Technology (AFIT) working in conjunction with Air Force Research Laboratories (AFRL) has coined the term “Computational Patterns of Life” (CPOL) to encompass research devoted to solving the problems posed by lack of automation when humans analyze intelligence data (including full motion video). CPOL is an umbrella term for the use of machines to explain past behaviors, issue real-time alerts, or predict future behavior across any domain where intelligence is gathered. Conceivable domains include text messages, email, Facebook and Twitter communications, telephone calls, photographs, video data of various resolution, and Hyperspectral Imagery (HSI). The chart in Figure 2.1 depicts one possible view of the CPOL family of solutions.

Referencing Figure 2.1, the areas labeled III, VI, and IX comprise AVS. This research focuses on areas labeled III and VI (highlighted), treating problems and technologies related to analyzing the past and providing real-time alerts using AVS.



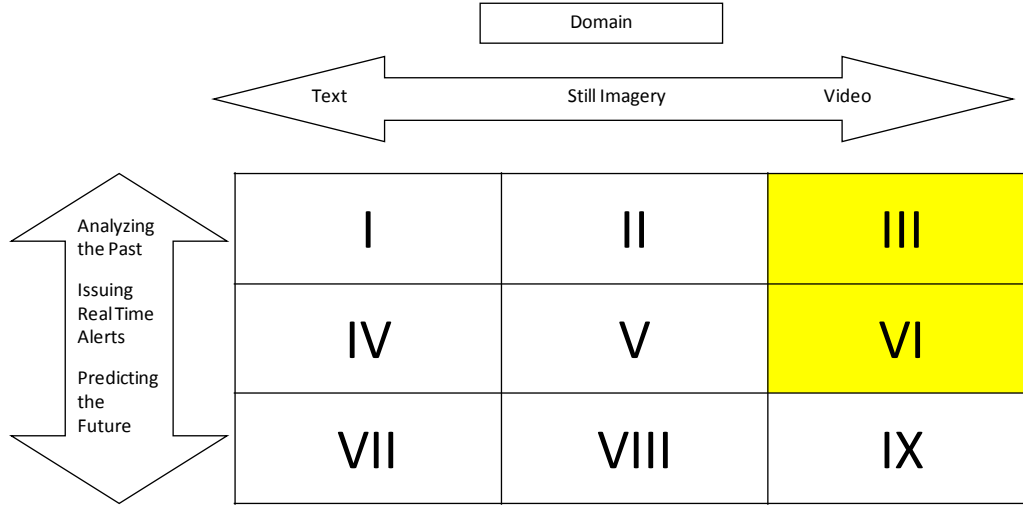


Figure 2.1: The CPOL family of solutions. This is one way of visualizing the various tasks and domains of interest.

### 2.1.1 Terminology.

Researchers use the term *visual surveillance* to describe any method for monitoring activity in an area and the term *Automated Visual Surveillance (AVS)* to describe the application of a machine to the task.

Although literature included in this survey gives no universal definition for the term, researchers typically use *Anomaly Detection* to mean the application of a machine to discover where observed behavior departs from common behavior [34]. This broad definition brings widespread agreement, but sharp divides exist in implementation details. Two major implementation approaches are found in literature. In the first (more commonly used) approach, the solution seeks to determine what behavior is normal and then to report any behavior outside that model [34, 39, 45, 54]. In the second approach, termed *a priori modeling*, specific definitions of what is anomalous are provided to the algorithm, which reports finding them [35].

Literature uses the terms *behavior decomposition* and *behavior modeling* to describe the act of a machine to derive patterns from the observed behaviors of the entities in a video sequence [7, 34]. For example, a machine might be trained to recognize a particular sequence of actions a human might take upon walking into a room—work at a desk, walk to the printer, then leave. *Duration modeling* is a form of behavior modeling primarily concerned with the length of time entities spend doing various actions [34]. Some authors use other terms to describe behavior modeling; Corbeil [5] uses the term *normalcy model* to indicate a numerical structure forming a baseline against which anomalous behavior can be judged. This thesis uses the terminology of *behavioral modeling* which produces a *behavioral model* against which to judge *anomalous behavior*.

A near-constant in implementations to solve CPOL problems is the application of a learning structure based upon graphs and feedback. Examples include Gaussian Mixture Models, Dynamic Bayesian Networks, and Artificial Neural Networks. We refer to them as *graphical structures* since that is the most common term in literature. Occasionally used is *Probabilistic Inference Network* which has the same meaning.

One difficult-to-describe concept is the moving parts of a scene an algorithm analyzes. Moving parts might be vehicles on a road, people in a parking lot, or employees in an office. Some papers refer to moving parts as *objects*, others as *blobs*. Some refer to moving images of interest as *players* within the scene. No term has won widespread acceptance. We adopt the term *entities* within this thesis.

### **2.1.2 State of the Art.**

Research into CPOL technology remains in its infancy. An exhaustive literature search has uncovered no operationally ready solutions. However, literature after 2005 shows a sharp spike in experiments and relative successes (see chronology in Figure 2.2). Still, most techniques treat only a part of the problem or have substantive work to be done before becoming operationally relevant.

A key distinction to make in the AVS techniques is that between *object based decomposition* and *behavior based decomposition*. This distinction represents the first division in classifying the myriad techniques designed to solve AVS problems.

*Object based decomposition* methods (also known as *tracking based methods*) use the process called tracking “to identify foreground pixels over time as belonging to a particular moving or stationary object” [7]. According to Dee [7] in her 2008 survey, almost all methods proposed to automate video surveillance depend on a hierarchy of techniques which may include tracking, scene modeling, behavior analysis, and detection of specific alarm events [7]. Each level in the hierarchy feeds data to a higher level. Thus, tracking is the most basic task, producing the data which all the others analyze. This data is usually described in a simple vector including only x position, y position, and velocity. A detected event in object based decomposition is one that a certain entity is moving in an unusual way. Most literature before 2006 reflects a strong tendency toward AVS methods that rely on tracking in some way.

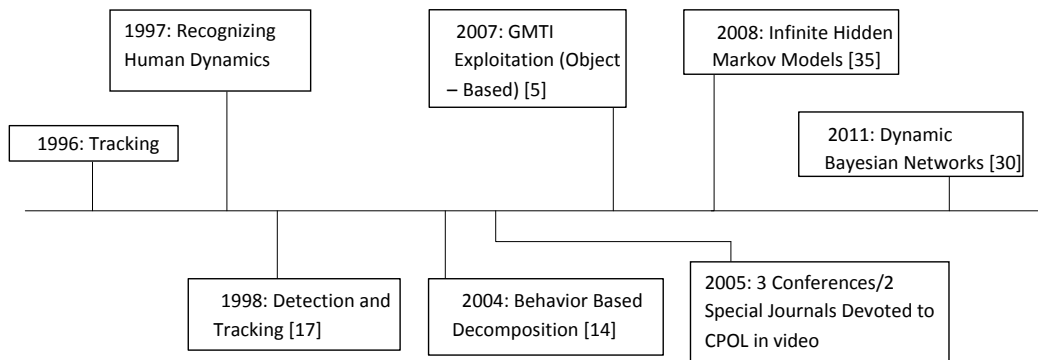


Figure 2.2: A timeline of innovation relevant to Automated Video Surveillance

However, since 2006 new methods have evolved which operate by analyzing more than just entity movement data; these are behavior based decomposition methods [33].

While object based decomposition detects anomalies in entity movement, behavior based decomposition detects anomalies by noting changes within the entire image. Note that even Behavioral Based Decomposition may perform some form of feature extraction to separate moving objects from background pixels. The vectors to which moving entities are reduced in behavior based decomposition are typically much larger than those in Object Based Decomposition, intending to capture more information about the image. The distinction between object and behavior based methods lies primarily in whether the method detects an anomaly primarily in the entities' movement or in a region of the image.

Although neither paradigm has yet yielded an operationally ready solution, both methods have seen some degree of success. A shortage of standardized methods of evaluation makes determining the best methods difficult [7, 55]. However, it is possible to observe how the state of the art has matured considering some of the later techniques described. While most methods claim an ability to detect an action of interest in some meaningful way, others have begun to tackle advanced considerations such as predicting future behavior [5, 29], ensuring a computable solution [34], or managing real-world concerns like multiple camera views [33].

### ***2.1.3 Object Based Decomposition Methods.***

This section describes related work in Object Based Decomposition, a method used in this research.

#### ***2.1.3.1 Experiments Involving Object Based Decomposition.***

Literature presents a wide range of experiments based upon tracking methods, most of them performed before 2005. This section describes four significant experiments.

Morris and Hogg [35] sum up the use of tracks to implement AVS the following way. The difficulty is reducing the motion data to a mathematical form from which statistical models can draw out anomalies. At closer resolutions, no two tracks will ever be performed in the exact same way. (To understand this concept, consider two people walking down a

hallway. At low resolutions, their paths may seem similar. However, when observing from a closer perspective, the chance their footsteps will land in the same places or that their lateral position in the hallway will be the same approaches zero.) Thus, naïve statistical models will consider all tracks anomalous against one another. Successful statistical models will determine which tracks are most like one another.

Morris and Hogg leverage this concept to detect behaviors common to car thieves in a crowded parking lot. To do this, they reduce points of interest to the entities' closest path of approach to other cars in the lot. An algorithm assigns each entity a probability on the assumption that low speeds and repeated approaches to cars are anomalous. The probabilities for a single entity over multiple cars are arranged into a sequence of increasing probability. The least probable terms are then compared with those of other entities to determine the most atypical trajectories. The method had good results in identifying anomalous behavior as portrayed by actors in the scene. All six entities acting like car thieves were identified. However, four normal trajectories were misclassified as anomalous, a fairly high misclassification rate. Also note that the method makes an assumption about what anomalous behavior of interest—someone looking for cars to break into. This is not a general solution, and it assumes that car thieves behave in a specific unchanging way [35].

Stauffer and Grimson [45] perform an experiment that realizes an interesting result using object based decomposition. Once they have tracked trajectories, they use a codebook of prototypes generated at run time (called on-line in machine learning parlance). This codebook initializes with a certain number of prototypes centered at existing data points. For each incoming object requiring classification, the classifier moves the prototypes closer to the object at some learning rate. This continues until it has a datastore permitting effective classification of tracked objects by identity (car, pedestrian, etc...) This data can then be used to determine which entities are anomalous. The determination has two inputs: the codebook data and comparison against co-occurrence data from like objects.

This idea relies on the idea the similar types of objects should behave similarly, which is of course a restrictive limitation the authors concede.

Oliver et al. [37] describe still another method based upon tracking. Their paper, published in 2000, is an early example of tracked entities being analyzed using a statistical Bayesian technique—in this case a form of Hidden Markov Model (HMM). Oliver et al. analyzed pedestrians in an outdoor scene to detect anomalies in the way two people respond to one another. An action was classified anomalous if the machine had never before seen a similar action. For example, prior explainable behaviors were “follow”, “approach, talk, and continue separately”, “approach, talk, continue together”, or “change direction, approach, talk, continue separately”. Their experiment demonstrated significant success in one common metric—a very high, flat Receiver Operating Characteristic (ROC) curve (see Section C.3.1).

Duong et al. [9] performed a similar experiment in that it relied on tracking and a statistical graphical model for analysis of video feed data. In this case, Duong attempted to automate the surveillance of elderly residents in assisted living apartments. The application is important, as computers may be able to save lives if they can alert medical personnel of unexplained stops in movement or strange behavior like strokes. The model used by Duong et al. is different from that used by Oliver et al.—a Switching Hidden Markov Model. Duong et al. also consider a domain different from Oliver et al.; Duong et al. seek to detect abnormal behavior in a single person whereas Oliver et al. detect abnormal behavior in two people’s response to one another. The Duong et al. experiment reports a high ROC curve [11], thus showing promise for operational application. At a 10% false positive rate the classifier achieves over 85% detection of abnormal conditions. This is not high enough for application to the intended domain (automated monitoring of elderly residents), but is high by comparison to other methods of 2005 and before.

The experiments described in this section were primarily performed before 2005. Later research efforts in automated video surveillance have moved toward behavioral based decomposition methods. However, some object based decomposition methods continue, primarily in defense-contracted activity.

## **2.2 Related work in Discrete or Single-Variable Hidden Markov Models**

Most related work in HMM application to AVS reduces model complexity by discretizing observations, quantizing multidimensional vectors into 1-D vectors, or both. This section detail several relevant examples of HMM use other than MOCDHMMs. Cited works are mostly from AVS or its parent fields (Computer Image Processing and Computer Vision). Examples from other fields are included only to show novel mathematical techniques. The work divides into studies which 1) reduce to 1-D observations, 2) discretize observations, and 3) do both.

### ***2.2.1 Related works which reduce to 1-D observations.***

In several image processing applications, researchers reduce multidimensional images to single-dimensional observation vectors by regarding slices of the image as individual 1-D observations. Several image classification algorithms reduce 2-D images to single-dimensional observation vectors by considering a block of the image at a time and inputting only a weighted mean grayscale value [4, 38, 44, 48]. Vstovsky and Vstovskaya [48] modify the Viterbi algorithm to produce the Network Algorithm (NA); NA discovers the method of traversing an image pixel by pixel that makes a particular HMM most likely to have produced the image. In this way, HMMs process 2-D images as series of 1-D observations.

Wilson and Bobick [50] perform work in gesture recognition; they reduce the multidimensional output PDF of various gestures to the smallest number of dimensions required to classify it by HMM. Wilson and Bobick's work lays a theoretical groundwork

for classification by multidimensional HMMs, but they perform experiments only in the case where the gesture can be reduced to a single dimension.

### ***2.2.2 Related works which discretize observations.***

A few papers reduce HMM complexity by discretizing the individual observations in N-dimensional vectors. This permits the use of discrete Probability Mass Functions (PMFs) vice continuous Probability Distribution Functions (PDFs).

Huang et al. [21] perform automated recognition of hand gestures; each finger is a variable in the 5-long observation vector. The positions of the fingers are discretized into the positions bending, half-bending, or straight.

Ham and Park [19] perform automated recognition of 3D objects in range images. Researchers multiply the real values of their feature vector by 30 (determined experimentally) and round to the nearest integer. Rounding each value within the feature vector to an integer ensures the values are discrete and capable of being modeled by a discrete PMF.

Rimey and Brown [43] explore automating camera motion in computer vision applications. Their feature vector is the  $(x, y)$  position of the fastest moving object in the frame; the variables  $x$  and  $y$  are “coarsely quantized”, but researchers do not give details on the quantization method they use.

### ***2.2.3 Related works which reduce to 1-D observations AND discretize observations.***

Some models in related works reduce training datasets to a series of 1-D, discrete symbols from a finite alphabet. Methods to achieve such a reduction include those relying on Vector Quantization (VQ) and those relying on methods other than VQ. This section first surveys 1-D, discretized methods using VQ and then survey a method not using VQ.

VQ has provided 1-D discrete observations to HMMs since the first application of HMMs in speech recognition in the late 1970’s. VQ reduces high-dimensional continuous vectors to 1-D discrete observations using a codebook of quantized regions. Each vector



is assigned to the quantized region it is closest to by some distance measure (usually Euclidean distance). The quantized regions are adjusted throughout the training procedure [51].

A large body of research leverages the VQ technique; examples of prominent work in speech recognition are [18, 32, 42]. Yamato [54] presents a similar case using VQ and mesh processing applied to recognizing human action in video data. Jung [26] applies the technique to hand gesture recognition. Forchhammer et al. [13] apply VQ observations and HMMs to lossless bi level image encoding.

Despite success in certain applications, VQ potentially subjects data to distortion if quantized regions are not able to be neatly partitioned. To overcome this limitation, variations have been proposed. Semi-Continuous HMMs supplement the VQ codeword with a continuous Gaussian variable representing how likely the VQ codeword is. Huang et al. [22] apply the technique to large-vocabulary speech recognition.

By contrast, Duong et al. [9] obtain 1-D, discrete symbols without using VQ. Duong et al. seek to detect anomalous behavior among elderly residents in assisted living in video data. Rather than devise a continuous feature vector, they divide the subject residents' rooms into 16 squares where each square number is the 1-D output symbol of their HMM. The Duong et al. experiment uses those integers from 1 to 16 as observations from what they call the Switching Semi-Hidden HMM (SSHMM).

### **2.3 Multi Observation Continuous Density Hidden Markov Models**

This chapter now turns to MOCDHMMs. Unlike studies surveyed above, MOCDHMMs do not use discretization of HMM output feature vectors. This section first describes why elimination of discretization may be helpful to HMM modeling and then surveys some related work using MOCDHMMs.

### ***2.3.1 Why MOCDHMMs are a good choice in Automated Visual Surveillance.***

MOCDHMMs are a good choice in AVS problems where minimum information loss from the feature processing step is required. Studies such as [19] and [21] perform well since the information loss in discretization does not impact separability. If this is the case, simpler observation vectors will likely produce comparable results to MOCDHMMs. (In our work, the dataset is difficult to separate, and more complex observation vectors are required.)

### ***2.3.2 Related work in MOCDHMMs.***

Only a small portion of research effort in AVS has been devoted to Multi Observation Continuous Density Hidden Markov Models. Most efforts that do use MOCDHMMs ignore possible dependency among variables, whereas the method of this thesis is sensitive to such dependency (also called covariance). Following is a survey of the few examples of MOCDHMM use reported in literature with specification of how our application differs.

The work presented in this thesis differs from previous MOCDHMM research in two major ways. First, by training HMMs that emit multi-dimensional observation vectors from non-diagonalized covariance matrices, we classify according to (among other things) covariance among the variables. In difficult-to-separate datasets, calculating such correlation among features may permit separating the classes. Second, we present a new method for selecting the correct HMM using scored evaluation rather than log-likelihood alone.

Tokuda et al. [46] present a theoretical framework for multi-space continuous HMM application without performing experimental evaluation of their method. They aim to solve problems presented when input data may contain discrete, 1-D observations and continuous, multi-dimensional observations simultaneously. The technique uses a generalized version of continuous-density Hidden Markov Models in which the length of the observation vectors is not constant. The model consists of a series of real spaces

$\Omega_1, \Omega_2, \dots, \Omega_G \in \Omega$ ; each real space may have a different dimension. Each real space has an associated probability that defines how likely it is to have contributed to the observation. Each observation consists of a set of indices describing which real spaces contributed and an  $n$ -dimensional vector of the continuous observations. In this way, a sample space can be designed to have a dimension of zero such that it is a single symbol. Certain applications cited by Tokuda et al. have datasets containing both multi-dimensional vectors as some observations and single-dimensional symbols as other observations; such varied dimensionality of observations within the same dataset makes the Tokuda framework relevant.

Vogler et al. [47] study automated recognition of American Sign Language (ASL), introducing the concept of the Parallel HMM (PaHMM). In this method, a different HMM is trained to model each component of the observation. Thus, eight HMMs are required to model the observation vectors of length eight in their experiment. Experimental evidence demonstrates that the method is effective for the ASL application. Note that because one HMM is used to model each observation in the feature vectors, no covariance among the observations is modeled in the Vogler et al. technique.

Fielding et al. [12] apply MOC DHMMs to recognition of moving light displays. They use six Fourier magnitude coefficients, one from each of six regions in the moving light displays. Lee et al. [31] apply a similar technique to refinement of speaker-independent speech-recognition databases; a 24-element vector is used as HMM output. However, Fielding et al. and Lee et al. constrain the covariance matrices of the Gaussian PDFs they use to be diagonal and ignore possible covariance among them. The method of this thesis relaxes the requirement to diagonalize the covariance matrices.

Pruteanu-Malinici and Carin [39] leverage the infinite HMM, an HMM with a theoretical infinite number of hidden states, to recognize abnormal events in closed circuit surveillance video. The feature vector used in [39] is 1200 parameters long. While

Pruteanu-Malinici and Carin do not specify that they use a non-diagonalized covariance matrix, any other formulation would require  $(1200)^2$  entries and is unlikely. Accordingly, we believe that the method of this thesis diverges from that of Pruteanu-Malinici since we allow non-diagonalized covariance matrices.

Examples of MOCDHMMs with non-diagonalized covariance matrices are in [3] and [16]. These two papers use the Gaussian PDFs without the generalization described in this thesis, apparently without encountering the same errors encountered in this thesis for reasons unspecified in the papers. It may be that certain datasets are more vulnerable to those errors.

## **2.4 Automated Video Surveillance: Conclusion**

This section has presented a literature survey concerning one subset of the CPOL family of problems, Automated Visual Surveillance (AVS). Research into AVS is still in its inception phase, and no solutions seem to be ready for operational fielding. However, since 2005 research work in the area has intensified. One of many available techniques for AVS involves Hidden Markov Models, most of which use some simplifying assumptions about output vectors. Such simplifications are Vector Quantization (VQ), discretizations of vector elements, techniques to reduce dimensionality to 1-D, or diagonalization of the covariance matrix. This thesis will describe a method to make HMM calculations without such simplifications.

### III. Methodology

This chapter describes the methods used to classify time-position tracks gathered from Full Motion Video (FMV). Classification is performed using Multi-Observation Continuous Density Hidden Markov Models (MOCDHMMs). The chapter first presents the approach taken to preprocess time-position tracks into observation sequences on which MOCDHMMs may operate. Second, the chapter presents the general approach for classification by Hidden Markov Model (HMM). It then provides the implementation details (for training and classification) of the general methodology under what we term the *standard MOCDHMM algorithm*. The chapter next details mathematical problems which result from using the standard MOCDHMM Algorithm and moves on to propose fixes to those problems under what we term the *modified MOCDHMM algorithm*. Finally, the chapter presents a methodology for empirically comparing the performance of the standard versus modified MOCDHMM algorithms.

#### 3.1 Preprocessing entity tracks in Full Motion Video (FMV)

Our method seeks to separate entities using track only, since that is the data expected to be available from a real-world sensor. *Preprocessing* is the process in Automated Visual Surveillance (AVS) which produces the vectors on which a mathematical structure will classify. For our purposes, preprocessing is used to generate observation sequences  $\mathcal{S} = (\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_T)$  from series of position and time data (time, x-coordinate, y-coordinate). One observation per timestep is recorded for each tracked entity from its source to its sink; the combined observations of one walk of one entity are termed a *sequence*. A group of sequences (usually believed to all have come from the same type of pedestrian) is a *sequence set*. See figure 3.1.

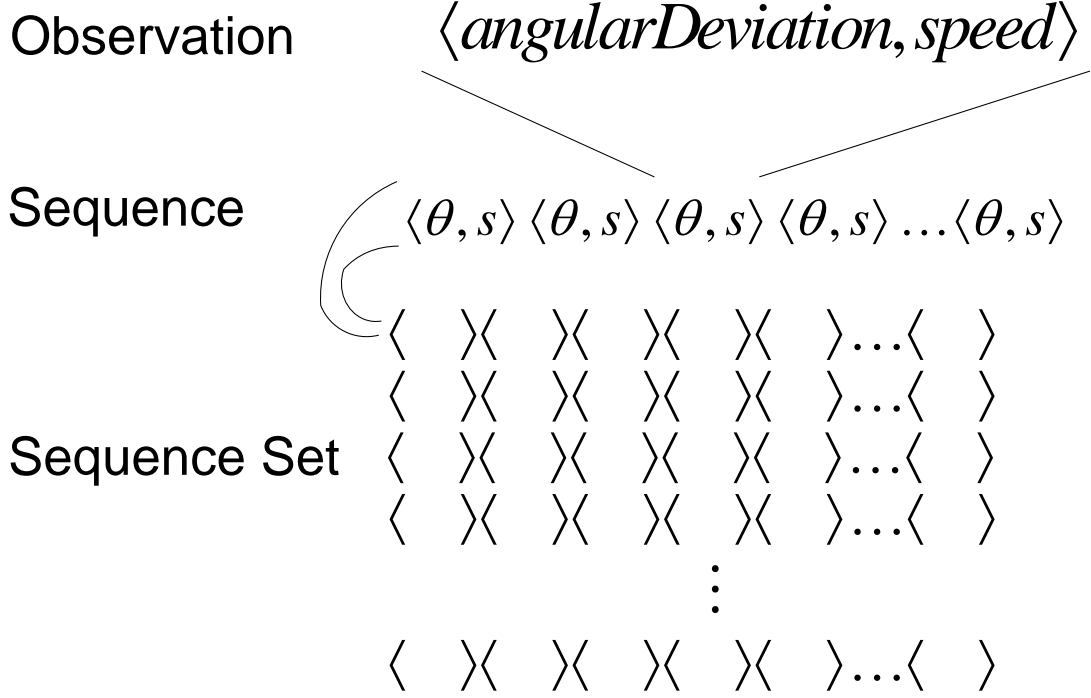


Figure 3.1: Definitions of observation, sequence, and sequence set. An observation is a state of one agent at one point in time. A sequence is the walk from source to sink of one agent. A sequence set is a group of sequences.

### 3.2 Classification by HMM

Algorithm 3.2.1 classifies observation sequences  $\mathcal{S}$  using HMMs. Both the standard MOCDHMM algorithm and the modified MOCDHMM algorithm follow the framework in Algorithm 3.2.1. Algorithm 3.2.1 takes three observation sequence sets: the class A training set  $\mathcal{E}^A$ , the class B training set  $\mathcal{E}^B$ , and a set  $\mathcal{E}^U$  of observation sequences whose class is unknown to the algorithm. (Researchers know the correct label for every sequence in  $\mathcal{E}^U$ .) Algorithm 3.2.1 returns a set of labels  $labels$  where  $labels[\mathcal{S}_i^U]$  is the label the algorithm assigns to sequence  $\mathcal{S}_i^U \in \mathcal{E}^U$ . In our experiments, the algorithm receives an accuracy ratio reflecting how often it is right versus the number of sequences it classified per Eq. (3.1).

$$\text{Classifier Accuracy} = \frac{\text{Correctly Classified}}{\text{Total Classified}} \quad (3.1)$$

In Algorithm 3.2.1 each sequence is classified according to the class  $X$  of whichever HMM  $\lambda_X$  has the highest likelihood of having produced the sequence. KMeansLearn takes a training set  $\mathcal{E}^X$  and a number of hidden states  $N$  and returns a trained HMM  $\lambda_X$  as described in Section 3.3.3. ForwardBackward returns the log-likelihood  $\mathcal{L}(\mathcal{S}_i^U | \lambda_X)$  of an HMM  $\lambda_X$  having produced sequence  $\mathcal{S}_i^U$  as described in Section 3.3.4. Algorithm 3.2.1 sets the label  $labels[\mathcal{S}_i^U]$  of each of each sequence  $\mathcal{S}_i^U$  to the class  $X$  which maximizes  $\mathcal{L}(\mathcal{S}_i^U | \lambda_X)$ .

**Algorithm 3.2.1:** CLASSIFYOBSERVATIONSEQUENCES( $\mathcal{E}^A, \mathcal{E}^B, \mathcal{E}^U$ )

```

 $\lambda_A \leftarrow \text{KMEANSLEARN}(\mathcal{E}^A, N)$ 
 $\lambda_B \leftarrow \text{KMEANSLEARN}(\mathcal{E}^B, N)$ 
for  $\mathcal{S}_i^U \in \mathcal{E}^U$ 
    do
         $\mathcal{L}(\mathcal{S}_i^U | \lambda_A) \leftarrow \text{FORWARDBACKWARD}(\mathcal{S}_i^U, \lambda_A)$ 
         $\mathcal{L}(\mathcal{S}_i^U | \lambda_B) \leftarrow \text{FORWARDBACKWARD}(\mathcal{S}_i^U, \lambda_B)$ 
         $labels[\mathcal{S}_i^U] \leftarrow \text{argmax}_{X \in \{A, B\}} \mathcal{L}(\mathcal{S}_i^U | \lambda_X)$ 
return ( $labels$ )

```

### 3.3 Standard MOCDHMM Algorithm

This section presents the *standard MOCDHMM Algorithm*. The algorithms and equations presented in this section are primarily taken from references [25, 41] which are theoretical treatments of HMMs published in 1989 and 1990 respectively. Although [25, 41] lay the theoretical groundwork for the work performed in this research on MOCDHMMs, they present no experimental validation of the techniques. Thus, they neither encounter nor treat several mathematical shortcomings which we have discovered to occur when applying MOCDHMMs.

This section presents general theory on discrete, single-observation HMMs, generalizes to theory concerning MOCDHMMs, and then presents two algorithms central to HMM use: Segmental K Means Training and the Forward Backward algorithm.

### 3.3.1 *Hidden Markov Models and the Markovian Assumption.*

HMMs are a generalization of Discrete Time Markov Chains (DTMCs). DTMCs are presented in this section to lay the groundwork for HMMs and their related algorithms described in Sections 3.3.2, 3.3.3, and 3.3.4.

A DTMC is a mathematical structure used to model stochastic processes. The structure is defined by a series of  $N$  states labelled  $\{1, 2, \dots, N\}$ . The model is always in some state  $i$  at time  $t$  and transitions according to a random probability to the next state (possibly back to  $i$ ). The transition probability is defined by a state transition matrix  $\mathbf{A}$  composed of  $a_{ij}$  which gives the probability of transitioning from state  $i$  to state  $j$  at time  $t + 1$ . The initial state is defined by a set of  $N$  elements  $\pi = \{\pi_1, \pi_2, \dots, \pi_N\}$  in which the probability of starting in state  $i$  is  $\pi_i$ . Then, the probability of realizing a particular state sequence  $s = (i_0, i_1, \dots, i_T)$  over  $T$  transitions is given by

$$\Pr(s|\mathbf{A}, \pi) = \pi_{i_0} \prod_{t=1}^T a_{i_{t-1}i_t} \quad (3.2)$$

Hidden Markov Models are a generalization of Discrete Time Markov Chains in which the state of the process is not directly observable. Instead, upon entry each state  $i$  emits an observation  $\mathbf{O}$  drawn from a Probability Mass Function (PMF)  $b_i(\mathbf{O})$  which gives the probability of emitting observation  $\mathbf{O}$  upon entering state  $i$ . HMMs thus produce an *observation sequence*  $\mathcal{S} = (\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_T)$  whereas DTMCs produce a *state sequence*  $s = (i_0, i_1, \dots, i_T)$ . The Hidden Markov Model is defined by a tuple of parameters  $\lambda = (\mathbf{A}, B, \pi)$ . In this tuple, the set  $B = \{b_1, b_2, \dots, b_N\}$  represents the PMFs of the hidden states. Using such a generalization renders Eq. (3.2) useless since it is no longer known



exactly which state has been entered at each time  $t$ . The probability of realizing observation sequence  $\mathbf{O}$  given HMM  $\lambda$  is then

$$\Pr(\mathbf{O}|\lambda) = \sum_i \pi_{i_0} \prod_{t=1}^T a_{i_{t-1}i_t} b_{i_t}(\mathbf{O}_t) \quad (3.3)$$

Note that, by these definitions, both Hidden Markov Models and Discrete Time Markov Chains obey the Markovian Assumption. That is, the future state of the process depends only on the present state and is independent of the past states. Note also that the analysis of this section assumes all observations  $\mathbf{O}$  are discrete; that is, they are drawn from a set of possible symbols in a finite alphabet  $V$  according to probabilities in a PMF. Moreover, the observations  $\mathbf{O}$  in this section are strictly one-dimensional. The next section generalizes to multi-observation continuous HMMs.

### ***3.3.2 Generalizing the HMM: Multi-Observation and Continuous Density Observations.***

Suppose that in any given state, an HMM need not emit an observation from a discrete PMF, but rather one from a continuous distribution. Then, the PMF  $b_i(\mathbf{O})$  becomes a Probability Density Function (PDF) instead. Furthermore, in 1-D HMMs described above, observations are constrained to a scalar values. However, in our research, we allow observations to be vectors of multiple values.

Thus, this thesis uses a generalized form of HMM referred to as a Multiple Observation Continuous Density HMM (MOCDHMM). MOCDHMMs require specialized training and evaluation procedures. Segmental K Means Training and Forward Backward Evaluation (Sections 3.3.3 and 3.3.4 respectively) work correctly for discrete, single-observation HMMs. However, both Segmental K Means Training and Forward Backward Evaluation in their unmodified forms cause mathematical errors when applied to MOCDHMMs. This section proceeds to give descriptions of Segmental K Means Training and Forward Backward Evaluation in their standard form. This section

details mathematical problems when applying Segmental K Means Training and Forward Backward Evaluation to MOCDHMMs in Section 3.4.

### 3.3.3 *Segmental K Means Training.*

Two methods exist for training a Hidden Markov Model: Baum Welch (BW) Training and Segmental K Means Training. Only Segmental K Means Training is used in this research. This section compares and contrasts BW and Segmental K Means Training in order to demonstrate how Segmental K Means Training works and motivate its use in this research.

K Means learning is useful because it is not highly sensitive to the choice of initial HMM parameters  $\lambda$ . By contrast, BW Training, being an Expectation Maximization technique, is susceptible to settling in suboptimal local minima depending on choice of initial HMM parameters  $\lambda$ . In addition, Segmental K Means is computationally simpler than BW Training.

Both BW Training and Segmental K Means Training use an iterative process; each iteration produces a new HMM  $\bar{\lambda}$  from an existing HMM  $\lambda$  such that  $\Pr(\mathcal{S}|\bar{\lambda}) > \Pr(\mathcal{S}|\lambda)$  for training observation sequence  $\mathcal{S}$ . However, BW Training and Segmental K Means Training use different methods to train the HMMs. BW Training uses Expectation-Maximization (E-M) to maximize the likelihood  $f(\mathcal{S}|\lambda)$  in Eq. (3.4).

$$f(\mathcal{S}|\lambda) = \sum_s \pi_{s_0} \prod_{t=1}^T a_{s_{t-1}s_t} \mathbf{b}_{s_t}(\mathbf{O}_t) \quad (3.4)$$

That is, BW Training maximizes by selection of HMM  $\lambda$  the likelihood of observing sequence  $\mathcal{S}$ . Thus, Eq. (3.4) is an objective function which sums over *all possible* state sequences.

By contrast, Segmental K Means Training uses a different optimization function than Eq. (3.4). Segmental K Means Training, unlike BW Training, seeks to maximize  $f(\mathcal{S}|\lambda)$

for the *most likely* state sequence. The Segmental K Means Learning objective function is shown in Eq. (3.5), where  $\max_s$  is a function which selects the most likely state sequence.

$$\max_s f(\mathcal{S}, s|\lambda) = \max_s \pi_{s_0} \prod_{t=1}^T a_{s_{t-1}s_t} \mathbf{b}_{s_t}(\mathbf{O}_t) \quad (3.5)$$

Accordingly, Segmental K Means Learning entails a two step process: 1) determine the most likely state sequence  $s$  for observation sequence  $\mathcal{S}$  and 2) optimize  $\lambda$  to make that  $s$  most likely given  $\mathcal{S}$ . Step one uses Viterbi decoding [41] to find the most likely state sequence. Step two determines a new HMM  $\bar{\lambda}$  to maximize Eq. (3.5). These two steps are continued until subsequent iterations produce no change between  $\lambda$  and  $\bar{\lambda}$ , a condition termed *convergence*. It can be shown that the algorithm will converge in all cases. For a proof of guaranteed convergence, see [25].

Whereas the Segmental K Means Algorithm produces an HMM  $\lambda$  trained to model a particular set of training data, it does not determine the likelihood  $f(\mathcal{S}|\lambda)$  of realizing a particular observation sequence  $\mathcal{S}$  from  $\lambda$ . To determine  $f(\mathcal{S}|\lambda)$ , the Forward Backward algorithm is used.

### 3.3.4 Forward Backward algorithm.

Let  $\lambda$  be an HMM trained (such as by Segmental K Means Training) to fit a particular set  $\mathcal{E}$  of training data. The forward-backward algorithm exists to determine the likelihood  $f(\mathcal{S}|\lambda)$  of realizing a particular observation sequence  $\mathcal{S}$  given  $\lambda$ . The naïve way to determine  $f(\mathcal{S}|\lambda)$  is to enumerate all possible state sequences  $s$  and determine the likelihood of realizing  $\mathcal{S}$  for each  $s$ . This is not computationally feasible, as there are on the order of  $O(N^T)$  values for  $s$  where  $N$  is the number of states and  $T$  is the length of the observation vector. To confront this problem, the Forward Backward Algorithm is used instead.

To illustrate the Forward Backward algorithm, consider the forward variable  $\alpha_t(i)$  as defined in Eq. (3.6).

$$\alpha_t(i) = f(\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_t, s_t = i | \lambda) \quad (3.6)$$

The forward variable  $\alpha_t(i)$  describes the probability of realizing the partial observation vector until time  $t$  and being in state  $i$  at time  $t$  given HMM  $\lambda$ . The algorithm to solve inductively for  $\alpha_t(i)$  is as follows [40]:

1. Calculate  $\alpha_1(i)$  given  $\pi_i$  and  $b_i$  for all hidden states  $i$

$$\alpha_1(i) = \pi_i b_i(\mathbf{O}_1), \quad 1 \leq i \leq N \quad (3.7)$$

2. For times  $t = 2, 3, \dots, T$  and hidden states  $i$ , calculate the forward variables  $\alpha_t(i)$

$$\alpha_{t+1}(i) = \left[ \sum_{j=1}^N \alpha_t(j) a_{ij} \right] b_i(\mathbf{O}_{t+1}) \quad (3.8)$$

3. Sum over the hidden states for the last timestep to determine the joint likelihood

$$f(S|\lambda) = \sum_{i=1}^N \alpha_T(i) \quad (3.9)$$

In this algorithm,  $b_i(\mathbf{O}_t)$  indicates the likelihood of observing  $\mathbf{O}_t$  given that the process is in state  $i$ . This algorithm is called the *forward algorithm*. (There also exists a backward algorithm for obtaining the same result. However, only the forward algorithm is used for this thesis.)

This simple form of the forward algorithm is inadequate for actual calculation, however, as long sequences underflow the likelihood calculation on most machines. Instead, a scaling procedure is used to directly calculate the log-likelihood. From Eq. (3.8), observe that each forward variable  $\alpha_t(i)$  is the sum of a large series of other forward variables  $\alpha_1$  through  $\alpha_{t-1}$  for each state. Each of these forward variables can be expressed by

$$\alpha_t(i) = \left( \prod_{\tau=1}^{t-1} a_{s_\tau s_{\tau+1}} \prod_{\tau=1}^t b_{s_\tau}(\mathbf{O}_\tau) \right) \quad (3.10)$$

where  $s_\tau$  is the state of the process at time  $\tau$ . In other words, each forward variable  $\alpha_t(i)$  is the result of repeated multiplication of probabilities  $a_{ij}$  and likelihoods  $b_j(\mathbf{O})$  for each timestep  $\tau < t$ . Note that each probability  $a_{ij}$  is less than one, each likelihood  $b_j(\mathbf{O})$  is usually less than one, and both  $a_{ij}$  and  $b_j(\mathbf{O})$  may be *much* less than one. Thus, the forward variable  $\alpha_t(i)$  approaches zero at an exponential rate as  $t$  increases and may result in machine underflow quickly. Accordingly, a scaling procedure is required to compute  $f(\mathcal{S}|\lambda)$ .

The scaling procedure modifies the forward algorithm of Eqs. (3.7) through (3.9). Using scaling, Eqs. (3.7) through (3.9) are replaced by Eqs. (3.11) through (3.15), where  $\hat{\alpha}_t(j)$  is the *scaled* forward variable,  $\omega_t(i)$  is the *intermediate* forward variable, and  $c_t$  is the *scaling factor*.

1. Initialize the intermediate forward variable for time 1

$$\omega_1(i) = c_1 \pi_i b_i(\mathbf{O}_1) \quad (3.11)$$

2. For each time  $t$  and hidden state  $i$ , compute the intermediate forward variable using the scaled forward variable  $\hat{\alpha}_{t-1}(i)$  for time  $t - 1$

$$\omega_t(i) = \sum_{j=1}^N \hat{\alpha}_{t-1}(j) a_{ji} b_i(\mathbf{O}_t) \quad (3.12)$$

3. Compute the scaling factor  $c_t$  by summing over the hidden states

$$c_t = \frac{1}{\sum_{j=1}^N \omega_t(j)} \quad (3.13)$$

4. For each hidden state  $i$ , compute the scaled forward variable at time  $t$  by multiplying the scaling factor by the intermediate forward variable. If  $t$  is not the last timestep, goto step 2.

$$\hat{\alpha}_t(i) = c_t \omega_t(i) \quad (3.14)$$

5. Determine the likelihood  $\mathcal{L}(S|\lambda)$  using only the scaling factors  $c_t$

$$\mathcal{L}(S|\lambda) = - \sum_{t=1}^T \ln c_t \quad (3.15)$$

According to Eqs. (3.11) through (3.14), each scaled forward variable is divided by the sum of the scaled forward variables across the hidden states for a given timestep. Rather than representing a likelihood directly, the scaled forward variable then represents what its *ratio* of likelihood is relative for that of the other states of the same timestep.

When using scaling, the likelihood  $f(S|\lambda)$  is not computed since it likely underflows the machine. Instead, the forward algorithm is modified to calculate only the log-likelihood  $\mathcal{L}(S|\lambda)$  of realizing a sequence. To make such a calculation, observe that  $\mathcal{L}(S|\lambda)$  is the negated sum of the logarithms of the scaling factors (Eq. (3.20)). To derive Eq. (3.20) we begin with the relationship between log-likelihood and likelihood in Eq. (3.16). We then follow a four-step process.

$$\mathcal{L}(S|\lambda) = \ln[f(S|\lambda)] \quad (3.16)$$

By Eq. (3.9) the likelihood  $f(S|\lambda)$  is the sum over the states for the *unscaled* forward variable  $\alpha_T(i)$  at the last timestep  $T$ . Substituting Eq. (3.9) in Eq. (3.16) gives Eq. (3.17).

$$\mathcal{L}(S|\lambda) = \ln \left[ \sum_{i=1}^N \alpha_T(i) \right] \quad (3.17)$$

Then, observe that the *scaled* forward variable  $\hat{\alpha}_t(i)$  equals the *unscaled* forward variable  $\alpha_t(i)$  multiplied by the product  $\prod_{t=1}^T c_t$  of all preceding scaling factors  $c_t$  (for a proof see Appendix B). Substituting  $\hat{\alpha}_t(i) = \prod_{t=1}^T c_t \alpha_t(i)$  into Eq. (3.17) gives (3.18).

$$\mathcal{L}(S|\lambda) = \ln \left[ \frac{1}{\prod_{t=1}^T c_t} \sum_{i=1}^N \hat{\alpha}_T(i) \right] \quad (3.18)$$

Then note that the sum  $\sum_{i=1}^N \hat{\alpha}_t(i)$  across the states of the *scaled* forward variable  $\hat{\alpha}_t(i)$  is one for all times  $t$  (for a proof see Appendix B). Substituting  $\sum_{i=1}^N \hat{\alpha}_t(i) = 1$  into Eq. (3.18) gives Eq. (3.19).

$$\mathcal{L}(\mathcal{S}|\lambda) = \ln \left[ \frac{1}{\prod_{t=1}^T c_t} \right] \quad (3.19)$$

Finally algebraically manipulate Eq. (3.19) to obtain Eq. (3.20) for calculating the log-likelihood  $\mathcal{L}(\mathcal{S}|\lambda)$  using scaling factors  $c_t$ , as desired. (See algebraic manipulation in Appendix B.)

$$\mathcal{L}(\mathcal{S}|\lambda) = - \sum_{t=1}^T \ln c_t \quad (3.20)$$

### 3.4 Mathematical problems with standard MOCDHMM algorithm

Applying Segmental K Means Training and the Forward Backward algorithm as described in literature [41] to MOCDHMMs causes several mathematical errors. This section details three errors that we have discovered to occur when Segmental K Means Training and the Forward Backward algorithm are applied to MOCDHMMs and shows analytically how those errors occur. The errors we treat are as follows: badly conditioned covariance matrices during training (Section 3.4.1), division by zero in Forward Backward Algorithm alpha scaling (Section 3.4.2), and biased classifiers (Section 3.4.3). Section 3.5 proposes fixes to the three errors to give what we call the *modified* MOCDHMM algorithm.

#### 3.4.1 Error 1: Badly conditioned covariance matrices.

This section first treats what we term *error 1*, which is badly conditioned covariance matrices produced during Segmental K Means training (Section 3.3.3).

Each hidden state in a MOCDHMM emits N-dimensional observations  $\mathbf{O}$  distributed around a multi dimensional Gaussian distribution given by Eq. (3.21), where  $\mu$  is the mean vector and  $\Sigma$  is the covariance matrix of the PDF  $b_i$  for hidden state  $i$ .

$$b_i(\mathbf{O}_t) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{O}_t - \mu)^T \Sigma^{-1}(\mathbf{O}_t - \mu)\right) \quad (3.21)$$

Eq. (3.21) requires a matrix inverse  $\Sigma^{-1}$ . If PDF  $b_i$  contains a near-singular covariance matrix  $\Sigma$ ,  $b_i(\mathbf{O}_t)$  cannot be determined by Eq. (3.21). Note that badly conditioned covariance matrices may arise at several points during Segmental K Means Training. To see why, consider that Segmental K Means trains HMMs iteratively; the HMM will have one covariance matrix  $\Sigma$  for each hidden state for each iteration. Because a near-singular  $\Sigma$  prevents Viterbi decoding (Section 3.3.3), a near-singular  $\Sigma$  in any hidden state at any iteration stops the training procedure.

#### 3.4.2 Error 2: Division by zero in alpha scaling.

We now turn to *error 2*, division by zero in alpha scaling. Error 2 occurs only in the Forward Backward Algorithm (Section 3.3.4).

Let  $\mathbf{w}_t \in \mathcal{S}^W$  be the observation at timestep  $t$  from the sequence  $\mathcal{S}^W$  generated by HMM  $\lambda_W$ . Consider calculating the log-likelihood  $\mathcal{L}(\mathcal{S}^W|\lambda_Z)$  of sequence  $\mathcal{S}^W$  given *different* HMM  $\lambda_Z$ . (The log-likelihood  $\mathcal{L}(\mathcal{S}^W|\lambda_Z)$  is required when classifying  $\mathcal{S}^W$  as class W or class Z.) Let  $b_i(\mathbf{w}_t)$  be the likelihood of realizing  $\mathbf{w}_t$  given that  $\lambda_Z$  is in state  $i$ . Suppose that  $\mathbf{w}_t$  is so unlikely for all PDFs in the hidden states of  $\lambda_Z$  that the probability underflows the machine for each state. That is,  $b_i(\mathbf{w}_t) \approx 0$  for  $i \in \{1, 2, \dots, N\}$ .

Then the forward algorithm will encounter a division by zero error due to Eqs. (3.12) and (3.13) as follows. In Eq. (3.22) (derived from Eq. (3.12)), the intermediate forward variable  $\omega_t(i) \approx 0$  for all hidden states  $i$  since the likelihood  $b_i(\mathbf{w}_t) \approx 0$  for all hidden states  $i$ . In Eq. (3.23) (derived from Eq. (3.13)),  $c_t$  at time  $t$  has a zero in the denominator.



$$\omega_t(i) = \sum_{j=1}^N \hat{\alpha}_{t-1}(j) a_{ji} b_i(\mathbf{w}_t) \quad (3.22)$$

$$= 0 \quad i \in \{1, 2, \dots, N\}$$

$$\begin{aligned} c_t &= \frac{1}{\sum_{i=1}^N \omega_t(i)} \\ &= \frac{1}{0} \end{aligned} \quad (3.23)$$

The calculation of  $\mathcal{L}(\mathcal{S}^W|\lambda_Z)$  is then undefined.

### 3.4.3 Error 3: Biased classifier.

The last error we treat, *error 3*, concerns biased classifiers. Errors 1 and 2 occur in Segmental K Means Training and the Forward Backward algorithm, respectively. By contrast, error 3 is a result of Segmental K Means Training and the Forward Backward algorithm working together to achieve classification.

An HMM classifier  $\Lambda$ , when classifying according to log-likelihood alone, may be *biased*. That is,  $\Lambda$  may select one class or the other too often, leading to classification accuracies well above random for one class and well below random for the other class.

To explain how a classifier may be biased, we first define the concepts of an HMM *recognizing* a sequence. Let  $\mathcal{S}^A$  and  $\mathcal{S}^B$  be arbitrary observation sequences from sequence sets  $\mathcal{E}^A$  and  $\mathcal{E}^B$  which trained HMMs  $\lambda_A$  and  $\lambda_B$  respectively. Let  $\Lambda_{AB}$  be the classifier formed by combining HMMs  $\lambda_A$  and  $\lambda_B$ . Then:

- An HMM *recognizes* the sequences it was trained on *better than* other sequences if and only if the average log-likelihoods of its own sequences is higher than others. That is, HMM  $\lambda_A$  recognizes sequence set  $\mathcal{E}^A$  better than sequence set  $\mathcal{E}^B$  if and only if the average log-likelihood  $\mathcal{L}(\mathcal{S}^A|\lambda_A) > \mathcal{L}(\mathcal{S}^B|\lambda_A)$ .

- A sequence set  $\mathcal{E}^A$  is *recognized* by an HMM  $\lambda_A$  *better than* another HMM  $\lambda_B$  if and only if its average log-likelihood is higher against  $\lambda_A$ . That is,  $\mathcal{E}^A$  is recognized by HMM  $\lambda_A$  better than  $\lambda_B$  if and only if on average  $\mathcal{L}(\mathcal{S}^A|\lambda_A) > \mathcal{L}(\mathcal{S}^A|\lambda_B)$ .
- Classifier  $\Lambda_{AB} = \{\lambda_A, \lambda_B\}$  is *biased toward* a sequence set  $\mathcal{E}^A$  if HMM  $\lambda_B$  recognizes  $\mathcal{E}^A$  better than does HMM  $\lambda_A$ .

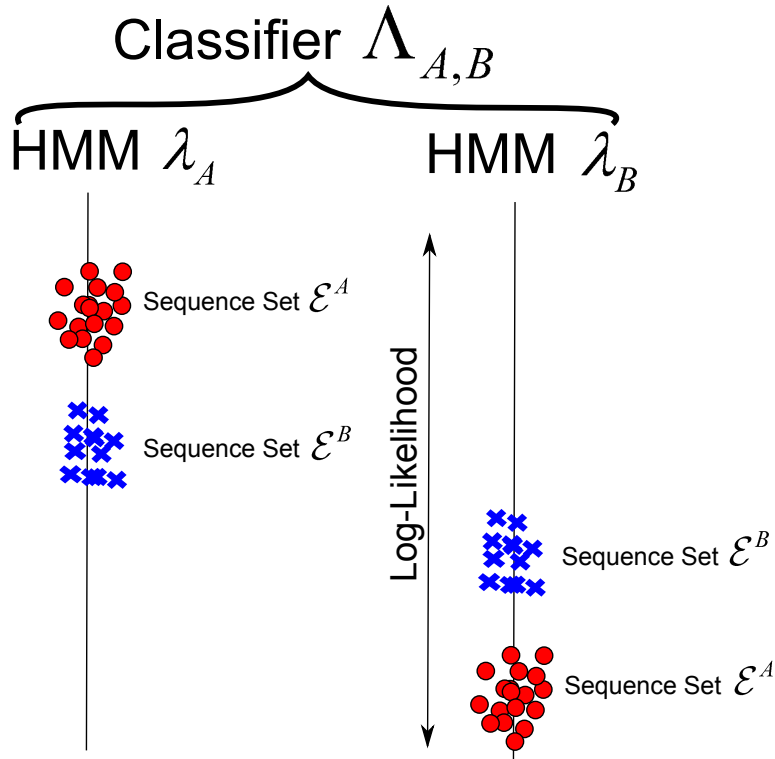


Figure 3.2: A biased classifier. In this depiction, the vertical axis represents log-likelihood  $\mathcal{L}(\mathcal{E}|\lambda)$ . The horizontal axis is meaningless; horizontal separation is allowed only to show multiple points close to one another on the vertical axis. It is expected that this classifier will produce substandard results. Since  $\mathcal{L}(\mathcal{E}^B|\lambda_A) > \mathcal{L}(\mathcal{E}^B|\lambda_B)$ , all  $\mathcal{E}^B$  will be classified as A

Figure 3.2 is a visual depiction of a biased classifier. Note that sequences in class B will be classified as class A most often because their log-likelihood against HMM A is higher.

### 3.5 Modified MOC DHMM algorithm

This section proposes fixes for each of the three errors given in Section 3.4 above. This section contains three subsections, one for each error described in Section 3.4.

#### 3.5.1 Proposed fix for error 1: Moore Penrose Pseudoinverse in Multivariate Gaussian PDF.

This section proposes a fix for error 1 (Section 3.4.1): badly conditioned covariance matrices. We refer to this fix as *fix 1*.

We propose changes to the Segmental K Means procedure for Gaussians proposed in the classic Rabiner tutorial [41]. The Segmental K Means in [41] calls for calculating the likelihood of an observation  $b_i(\mathbf{O})$  based on the *degenerate* Gaussian PDF given in Eq. (3.21). However, the covariance matrix inverse  $\Sigma^{-1}$  causes problems during training of MOC DHMMs as detailed in Section 3.4.1. To address this problem, we change Eq. (3.21) used to calculate  $b_i(\mathbf{O})$  so that it does not require a matrix inverse. Thus, even a badly-conditioned covariance matrix  $\Sigma$  can produce a valid likelihood  $b_i(\mathbf{O})$ .

The new formula for  $b_i(\mathbf{O})$  is adopted from Multivariate Gaussian theory. Such theory prescribes a *generalized* Gaussian PDF  $f(x)$ , defined in Eq. (3.24) [10]. In Eq. (3.24),  $\Sigma$  is the covariance matrix,  $\mu$  is the mean vector,  $\text{rank}()$  returns the rank of the matrix,  $\Sigma^-$  denotes the generalized inverse, and  $\det^*$  denotes the pseudo-determinant.

$$f(x) = \frac{1}{\sqrt{(2\pi)^{\text{rank}(\Sigma)} \det^*(\Sigma)}} \text{Exp} \left[ -\frac{1}{2} (x - \mu)^T \Sigma^- (x - \mu) \right] = b_i(\mathbf{O}) \quad (3.24)$$

The concepts from Eq. (3.24) which do not exist in the degenerate Gaussian likelihood formula (Eq. (3.21)) are as follows: rank, pseudo-determinant, and generalized inverse. These three concepts are treated separately in the next paragraphs.

**Rank.**  $\text{rank}(\Sigma)$  is the number of linearly independent rows in  $\Sigma$ . If  $n \times n$  matrix  $\Sigma$  has  $\text{rank}(\Sigma) = n$ , it is said to be *full rank* and always has determinant  $|\Sigma| > 0$ . Thus, a full-rank  $\Sigma$  always has a matrix inverse  $\Sigma^{-1}$ . However, a full rank  $\Sigma$  might still have determinant  $|\Sigma|$  close to zero and thus be difficult to invert [10].

**Pseudo-determinant.** The pseudo-determinant ensures that even matrices whose determinants are zero to machine precision are still supported by the PDF. Note that for the degenerate Gaussian PDF in Eq. (3.21),  $|\Sigma|$  is in the denominator of a fraction. Thus, the degenerate Gaussian PDF is undefined if  $|\Sigma| = 0$ . The pseudo-determinant  $\det^*(\Sigma)$  in the generalized Gaussian PDF replaces the determinant  $|\Sigma|$  in the degenerate Gaussian PDF. The formula for a pseudodeterminant  $\det^*(\Sigma)$  of the  $n \times n$  matrix  $\Sigma$  is in Eq. (3.25) [10]. Note that in Eq. (3.25), the pseudo-determinant  $\det^*(\Sigma)$  reduces to the standard determinant  $|\Sigma|$  if  $\Sigma$  is full rank.

$$\det^*(\Sigma) = \lim_{\alpha \rightarrow 0} \frac{|\Sigma + \alpha \mathbf{I}|}{\alpha^{n - \text{rank}(\Sigma)}} \quad (3.25)$$

**Generalized inverse.** The inverse  $\Sigma^{-1}$  of a matrix  $\Sigma$  satisfies  $\Sigma \Sigma^{-1} = \mathbf{I}$  where  $\mathbf{I}$  is the identity matrix. The *generalized* inverse  $\Sigma^-$  of matrix  $\Sigma$  satisfies the two less stringent requirements in Eqs. (3.26) and (3.27) [2]:

$$\Sigma \Sigma^- \Sigma = \Sigma \quad (3.26)$$

$$\Sigma^- \Sigma \Sigma^- = \Sigma^- \quad (3.27)$$

Several procedures to calculate the pseudoinverse are available. Although any matrix satisfying Eqs. (3.26) and (3.27) is an acceptable matrix for the generalized multivariate Gaussian PDF in Eq. (3.24), the research of this thesis adopts the Moore Penrose Pseudoinverse because it produces a unique generalized matrix inverse  $\Sigma^-$  for every  $\Sigma$ . The Moore Penrose Pseudoinverse satisfies the following two properties in Eqs. (3.28) and (3.29), in addition to those given in Eqs. (3.26) and (3.27):

$$(\Sigma \Sigma^-)^* = \Sigma \Sigma^- \quad (3.28)$$

$$(\Sigma^- \Sigma)^* = \Sigma^- \Sigma \quad (3.29)$$

In Eqs. (3.28) and (3.29),  $\Sigma^*$  is the conjugate transpose of  $\Sigma$ . In this research Singular Value Decomposition (SVD) method of calculating the Moore Penrose pseudoinverse is adopted. Thus, if matrix  $\Sigma$  decomposes by SVD such that  $\Sigma = \mathbf{U} \mathbf{A} \mathbf{V}^*$  then the Moore Penrose pseudoinverse  $\Sigma^- = \mathbf{V} \mathbf{A}^- \mathbf{U}^*$ . Note that by SVD,  $\mathbf{A}$  is always diagonal, and its pseudoinverse  $\mathbf{A}^-$  can be taken by inverting all its nonzero elements and transposing the matrix [2].

### 3.5.2 Proposed fix for error 2: Return negative infinity if the scaling factor is zero.

This section proposes a fix for error 2 (Section 3.4.2): division by zero in the alpha scaling procedure. We refer to this fix as *fix 2*.

In this proposed modification, the piecewise function in Eq. (3.30) replaces Eq.(3.13) for the Forward Backward algorithm.

$$c_t = \begin{cases} \frac{1}{\sum_{j=1}^N \omega_t(j)} & , \quad \sum_{j=1}^N \omega_t(j) > 0 \\ \infty & , \quad \sum_{j=1}^N \omega_t(j) \approx 0 \end{cases} \quad (3.30)$$

In this way, problem calculations for the log-likelihood  $\mathcal{L}(S|\lambda)$  no longer produce division by zero errors during  $c_t$  calculation in Eq. (3.13). To see why, suppose  $\sum_{j=1}^N \omega_t(j) = 0$ . By Eq. (3.13), calculating  $c_t$  yields a division by zero error; by Eq. (3.30),  $c_t$  is set to infinity. If  $c_t$  is set to infinity then  $\sum_{t=1}^T c_t = \infty$  as well. Thus, using Eq. (3.20),  $\mathcal{L}(S|\lambda) = -\infty$ , indicating that HMM  $\lambda$  is so unlikely to have produced  $S$  that  $\lambda$  should not be considered when classifying  $S$ . Because the main classification algorithm (Algorithm 3.2.1) classifies sequences according to which HMM gives the maximum log-likelihood, HMM  $\lambda$  will not be considered. Furthermore, no division by zero errors result.

### 3.5.3 Proposed fix for error 3: Use scored evaluation instead of raw log-likelihood.

This section proposes a fix for error 3 (Section 3.4.3): biased classifiers. We refer to this fix as *fix 3*.

We contend with the biased classifier problem by defining a new type of HMM, the scored HMM  $\hat{\lambda}$ . Let  $\hat{\lambda}$  include the same tuple of values that defines a standard HMM  $\lambda$ : the transition matrix  $\mathbf{A}$ , the set of PDFs  $B$ , and the starting probabilities  $\pi$ . In addition to  $\lambda$ ,  $\hat{\lambda}$  includes two additional parameters: mean log-likelihood  $\hat{\mu}$  and standard deviation of log-likelihood  $\hat{\sigma}$ . To find  $\hat{\mu}$  and  $\hat{\sigma}$ , we calculate a set of log-likelihoods  $L$  with one log-likelihood  $\mathcal{L}(S|\lambda)$  for each sequence  $S$  in the training data  $\mathcal{E}$ . We set  $\hat{\mu}$  and  $\hat{\sigma}$  equal to the mean and standard deviation of  $L$  respectively. The training procedure yielding  $\hat{\lambda}$  from a set of training sequences  $\mathcal{E}$  is in Algorithm 3.5.1.

Given a trained scored HMM, we obtain a score  $z(S|\hat{\lambda})$  (vice a raw log-likelihood  $\mathcal{L}(S|\lambda)$ ) for sequence  $S$  using  $z(S|\hat{\lambda}) = \frac{\mathcal{L}(S|\lambda) - \hat{\mu}}{\hat{\sigma}}$ . A score thus removes the bias from a classifier by comparing against the distribution of log-likelihoods  $L$  rather than log-likelihood alone. The classifier then classifies  $S$  according to its score yielded by Algorithm 3.5.2 rather than  $\mathcal{L}(S|\lambda)$  as indicated in Algorithm 3.2.1.

#### Algorithm 3.5.1: TRAINSCOREDHMM( $\mathcal{E}^A$ )

```

 $\lambda \leftarrow \text{KMEANSLEARN}(\mathcal{E}^A, N)$ 
 $(\mathbf{A}, B, \pi) = \lambda$ 
 $L \leftarrow \emptyset$ 
for  $S \in \mathcal{E}$ 
  do  $\begin{cases} \mathcal{L} \leftarrow \text{FORWARDBACKWARD}(S, \lambda) \\ L.\text{ADDTOSET}(\mathcal{L}) \end{cases}$ 
 $\hat{\mu} \leftarrow \text{MEAN}(L); \hat{\sigma} \leftarrow \text{STANDARDDEVIATION}(L)$ 
 $\hat{\lambda} \leftarrow (\mathbf{A}, B, \pi, \hat{\mu}, \hat{\sigma})$ 
return  $(\lambda)$ 

```

**Algorithm 3.5.2:** SCOREDFORWARDBACKWARD( $\mathcal{S}, \hat{\lambda}$ )
$$\left\{ \begin{array}{l} (\mathbf{A}, B, \pi, \hat{\mu}, \hat{\sigma}) \leftarrow \hat{\lambda} \\ \lambda \leftarrow (\mathbf{A}, B, \pi) \\ \mathcal{L} \leftarrow \text{FORWARDBACKWARD}(\mathcal{S}, \lambda) \\ \text{return } (\frac{\mathcal{L} - \hat{\mu}}{\hat{\sigma}}) \end{array} \right.$$

Algorithm 3.5.3 is a modified version of Algorithm 3.2.1 incorporating the fixes of this section. In Algorithm 3.5.3, KMeansLearn is replaced by TrainScoredHMM, ForwardBackward is replaced by ScoredForwardBackward, and  $\mathcal{L}(\mathcal{S}_i^U | \lambda_B)$  is replaced by  $z(\mathcal{S}_i^U | \hat{\lambda}_A)$ .

**Algorithm 3.5.3:** MODIFIEDCLASSIFYOBSERVATIONSEQUENCES( $\mathcal{E}^A, \mathcal{E}^B, \mathcal{E}^U$ )
$$\left\{ \begin{array}{l} \lambda_A \leftarrow \text{TRAINSCOREDHMM}(\mathcal{E}^A, N) \\ \lambda_B \leftarrow \text{TRAINSCOREDHMM}(\mathcal{E}^B, N) \\ \text{for } \mathcal{S}_i^U \in \mathcal{E}^U \\ \quad \left\{ \begin{array}{l} z(\mathcal{S}_i^U | \hat{\lambda}_A) \leftarrow \text{SCOREDFORWARDBACKWARD}(\mathcal{S}_i^U, \lambda_A) \\ \text{do } \left\{ \begin{array}{l} z(\mathcal{S}_i^U | \hat{\lambda}_B) \leftarrow \text{SCOREDFORWARDBACKWARD}(\mathcal{S}_i^U, \lambda_B) \\ labels[\mathcal{S}_i^U] \leftarrow \arg\max_{X \in \{A, B\}} z(\mathcal{S}_i^U | \hat{\lambda}_X) \end{array} \right. \end{array} \right. \\ \text{return } (labels) \end{array} \right.$$

### 3.6 Experimental Design

Three experiments are conducted to experimentally demonstrate the effectiveness of problem fixes proposed in Section 3.5. All experiments in this research relate to Automated Visual Surveillance (AVS) in overhead full motion video (FMV). The first experiment is a pilot experiment using data generated by Gaussian Mixture Models (GMMs); the dataset in the pilot experiment uses multi-observation continuous vectors designed to be easy to separate. The second experiment uses the same dataset as the first; however, it applies a *blending* procedure to characterize the MOC DHMM classifier's response to errors in

training data. The third experiment uses a simulation of an overhead FMV scene consisting of soldiers in a combat environment walking from source locations to sink locations.

All experiments apply the standard MOCDHMM algorithm from Section 3.3 to arrive at a baseline for performance. All experiments then apply the modified MOCDHMM algorithm from Section 3.5 to allow relative increase in performance. Both experiments are implemented using JAHMM [14], and open-source Java HMM package.

To standardize terminology, we use the following terms in both experiments: *observation*, *sequence*, and *sequence set*. An *observation*  $\mathbf{O}$  is the condition (expressed in some numerical term) of one entity at one point in time (sampled at 1 Hz). A *sequence*  $\mathcal{S}$  is a series of observations representing the walk from source to sink of one entity. A *sequence set*  $\mathcal{E}$  is a group of sequences.

### **3.6.1 Unblended Pilot Experimental Design.**

In order to generate agents which are easy to separate, Gaussian Mixture Models (GMMs) are used. GMMs employ a series of Gaussian distributions. Each Gaussian Distribution has a certain probability (its *mixture*) of producing the next observation. Three Gaussian Mixture Models were used in this experiment: Small Variation Gaussian (SVG), Wide Variation Gaussian (WVG), and Widely Separated Mixed Gaussian (WSMG). Table 3.1 gives the specific mixtures, means, and standard deviations of the various distributions used in the pilot experiments.

The pilot experiment aims to approximate the complexity of the observations obtained in tracked FMV. We use observation vectors of length two in order to allow for two variables extracted from tracked FMV (such as position and speed). Under our construct, the two values in each observation vector are independent of one another. SVG, WVG, and WSMG are used in all combinations of each of the two variables in the observation vectors, yielding nine agents labelled A through I. Table 3.2 defines the agent types. (Note that the two



Table 3.1: GMM Definitions: the three building blocks of the agents in both pilot experiments

Distribution Type		Mixture	Mean	Standard Deviation
SVG	Gaussian 1	1.0	$\mu_s = 6$	$\sigma_s = 0.2$
	Gaussian 2	0		
WVG	Gaussian 1	1.0	$\mu_w = 6$	$\sigma_w = 6$
	Gaussian 2	0		
WSMG	Gaussian 1	0.5	$\mu_{m1} = 2$	$\sigma_m = 1$
	Gaussian 2	0.5	$\mu_{m2} = 10$	$\sigma_m = 1$

variables in the observation vectors are arbitrary and correspond to no actual or simulated data in FMV. The two variables are labelled  $\Theta$  and  $S$ .)

Table 3.2: Names of the nine simulated agents for the pilot experiments

		Var 2: $S$		
		WVG	SVG	WSMG
Var 1: $\Theta$	WVG	A	B	C
	SVG	D	E	F
	WSMG	G	H	I

In the pilot experiments, HMMs always have two hidden states. Accordingly, the HMM can be fully qualified by Eqs. (3.31a) through (3.31d), in which  $\mathbf{A}$  is the  $2 \times 2$  state transition matrix,  $\pi_i$  is the probability of beginning the process in state  $i$ ,  $\mathbf{b}$  is a vector of distributions where  $b_i$  is the observation emission distribution for state  $i$ , and  $k$  is the dimensionality of the observation vector ( $k = 2$  for the pilot experiments). The means in state  $i$  are  $\mu_{i\Theta}$  and  $\mu_{iS}$  for variables 1 and 2 respectively; the variances in state  $i$  are  $(\sigma_{i\Theta})^2$

and  $(\sigma_{iS})^2$  for variables 1 and 2 respectively. The covariance between variables 1 and 2 in state  $i$  is  $Cov_i[\Theta, S]$ .

$$\lambda = (\pi, \mathbf{A}, \mathbf{b}) \quad (3.31a)$$

$$b_i \sim \mathcal{N}_k(\mu_i, \Sigma_i) \quad (3.31b)$$

$$\mu_i = [\mu_{i\Theta} \mu_{iS}] \quad (3.31c)$$

$$\Sigma_i = \begin{bmatrix} (\sigma_{i\Theta})^2 & Cov_i[\Theta, S] \\ Cov_i[\Theta, S] & (\sigma_{iS})^2 \end{bmatrix} \quad (3.31d)$$

We expect that the best way to model agents with no occurrences of WSMG distributions is using one state. That is, the best model for the observations of agent D(i.e. [SVG, WVG]) is as follows

$$b_0 \sim \mathcal{N}_2 \left( [\mu_s \mu_w], \begin{bmatrix} \sigma_s^2 & 0 \\ 0 & \sigma_w^2 \end{bmatrix} \right) \quad (3.32)$$

Here,  $b_0$  is the random vector of two observations ( $\Theta$  and  $S$ ) for agent D.  $\mathcal{N}_2$  denotes the two dimensional multivariate Gaussian distribution. The HMM should model a process which arrives in one state and stays there. In this case the A matrix should be

$$\mathbf{A}_D = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

By contrast, agent F requires two multivariate Gaussians to model correctly:

$$b_0 \sim \mathcal{N}_2 \left( [\mu_s \mu_{m1}], \begin{bmatrix} \sigma_s^2 & 0 \\ 0 & \sigma_m^2 \end{bmatrix} \right) \quad (3.33a)$$

$$b_1 \sim \mathcal{N}_2 \left( [\mu_s \mu_{m2}], \begin{bmatrix} \sigma_s^2 & 0 \\ 0 & \sigma_m^2 \end{bmatrix} \right) \quad (3.33b)$$

In this case, we expect the following A matrix since the mixture between the two Gaussians is 0.5.

$$\mathbf{A}_F = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

Note that under this construct, there is no effective way to model agent I with fewer than four hidden states.

HMMs in the pilot experiments are trained with 1000 sequences of 50 2-D observations each. The classifiers are tested and assigned accuracy ratings based on their classification performance of 2000 sequences; that is, the sequence set of unknown sequences consists of 1000 sequences from each class used to train the HMMs. Both pilot experiments are pairwise experiments; the classifiers in the pilot experiments attempt to classify only two agents at a time. To that end, the classifiers contain only two HMMs, one HMM trained on each class to classify.

### **3.6.2 Blended Pilot Experimental Design.**

The blended pilot experiment stresses the mathematical classifier in an attempt to find out how it is most likely to fail. One common problem the classifier is likely to face on a real battlefield is the input of mislabeled training sequences. That is, not all training data may be correctly labelled by human analysts. Thus blending procedure is performed on the pilot test datasets. In *blending*, a number of sequences within the training sequence sets are replaced by sequences from a different agent. For example, we denote agent A sequences of which 30% have been replaced by agent B sequences as the “AblendB30” dataset. The “AblendB30” dataset was classified against the “BblendA30” dataset. In this example, the 30% is termed *blend density*. Intuitively, classification problems with higher blend densities are more difficult to classify since the training data provided to the classifier is more confused. See Figure 3.3 for a visual depiction of the Blended Pilot Experiment.

The Blended Pilot Experiment uses the same number of sequences and the same sequence length as in the Unblended Pilot Experiment (Section 3.6.1). Each HMM is trained on 1000 sequences. Each classifier contains two HMMs; the classifier is tested against 2000 **unblended** sequences whose labels are unknown to the classifier.

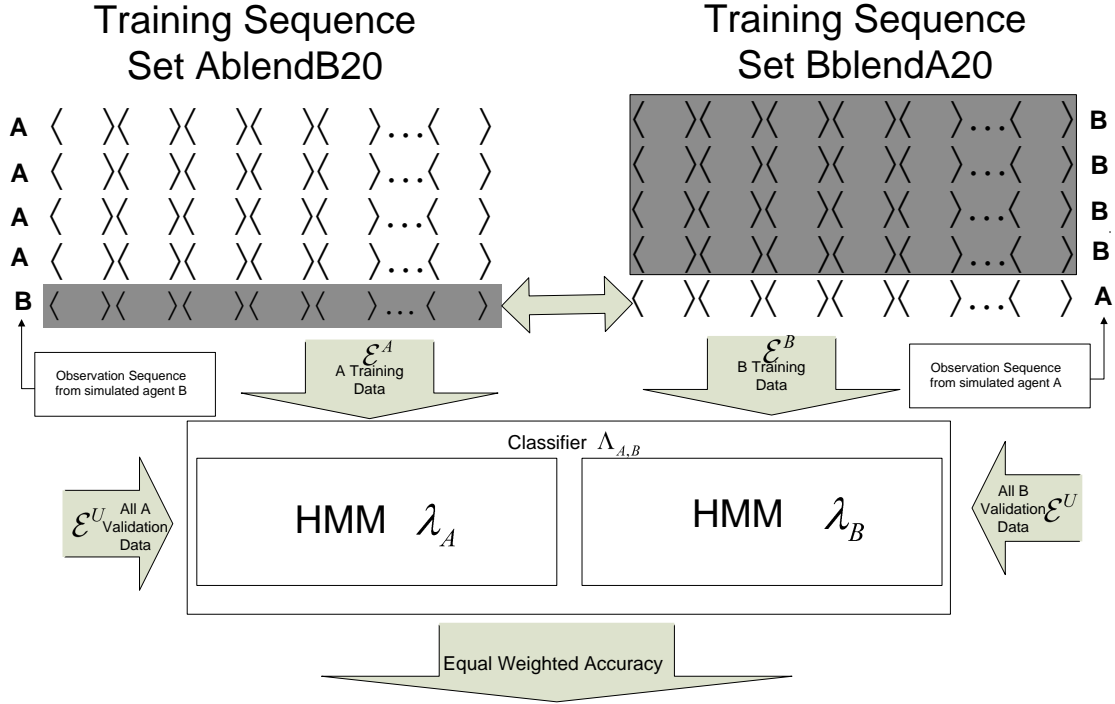


Figure 3.3: Sequence mixed datasets used in Blended Pilot Experiment

We then characterize the results according to one of three profiles for error resistance: Error Vulnerable, Linear Response, or Error Resistant. An *Error Vulnerable* classifier would cause a rapid loss of accuracy at a small blend density (sequence mis-labeling). A *Linear Response* classifier would experience about the same loss of classification accuracy as blend density. An *Error Resistant* classifier would experience small decreases in classifier accuracy given initial increases in blend density. See Figure 3.4 for a depiction of the possible error responses.

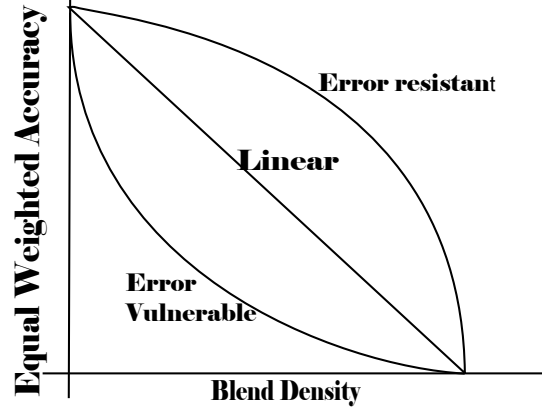


Figure 3.4: Error response profiles for characterization of error resistance in Blended Pilot Experiment

We hypothesize that the MOCDHMM classifier will exhibit error resistance, as one strength of traditional HMMs is appropriate response to noisy data [41].

### 3.6.3 *Fryer Dataset Experimental Design.*

A simulated overhead combat scenario containing moving pedestrians is used to demonstrate the effectiveness of the modified MOCDHMMs. A dataset whose training data is difficult to separate by track alone is ideal for this purpose. Thus the *Fryer Dataset* [15] is adopted, which contains entities who act very similarly to one another. The Fryer Dataset entities are recorded from the movements of artificially intelligent agents modeled by the JACK Intelligent Agent Framework [17]. Five classes of artificially intelligent agents are used: AWARE, CARELESS, SAFE, COMBAT, and STEALTH. The various behaviors are defined in previous research [15] by the SimExec agent framework operating on the Visual Battlespace 2 (VBS2) combat simulator. High-level features of the agent classes are as follows:

- CARELESS: Does not engage enemy targets unless directly fired upon. Maintains straight path toward goal otherwise

- SAFE: Moves as per CARELESS, except will assume AWARE behavior when encountering the enemy
- AWARE: Assumes moderate (as opposed to full) speed, using cover when available
- COMBAT: Will engage any detected enemy, will fire while moving, and may move while prone
- STEALTH: Spends most time crawling, except will move quickly to cross open terrain into greater concealment

The dataset is difficult to separate since entities behave similarly in the absence of prominent terrain features and multiple enemy actors.

The Fryer dataset contains 1997 labelled tracks of each class of agent from its origin to its destination: CARELESS, SAFE, AWARE, COMBAT, STEALTH. Each agent track is sampled one time per second. The number of samples for each track varies because agents take a different amount of time to reach their destinations. On average the agents take 50 seconds to reach their destinations and the tracks consist of 50 samples. Each sample contains the time followed by the agent's x-coordinate, y-coordinate, elevation, and facing direction in degrees. Because real-world technologies for determining elevation and facing direction in overhead FMV are less mature than those for determining position, only time, x-position, and y-position are considered in this research.

For each agent, the 1997 labelled tracks of approximately 50 samples each are preprocessed to 1997 sequences of approximately 50 2-D observations each. Thus, obtain five sequence sets are obtained:  $\mathcal{E}^{\text{CARELESS}}$ ,  $\mathcal{E}^{\text{SAFE}}$ ,  $\mathcal{E}^{\text{AWARE}}$ ,  $\mathcal{E}^{\text{COMBAT}}$ , and  $\mathcal{E}^{\text{STEALTH}}$ .

To test the accuracy of the MOCDHMM classifiers, we perform pairwise classification. MOCDHMM classifiers perform classification of each pair of datasets. Thus, the five datasets require 10 test cases to ensure all datasets are classified against all other datasets. For each test case, a common machine learning methodology called cross validation is ap-

plied to ensure training data is never used as testing data . Under  $q$ -fold *cross validation*, the train-test cycle in each test case is performed  $q$  times; each cycle is called a *fold*. In each fold a number of training sequences is withheld from training and used for testing instead. In general, the number of sequences withheld is  $\frac{L}{q}$ , where  $L$  is the number of sequences in the dataset. In this way, each sequence is used as testing data exactly once, and no sequence is ever used for both training and testing in the same fold.

### 3.6.3.1 Preprocessing the Fryer dataset.

We design a model for analyzing the Fryer dataset that takes into account two features:

- 1) angular deviation  $\Theta$  in degrees from average direction of movement, and 2) speed  $S$ .

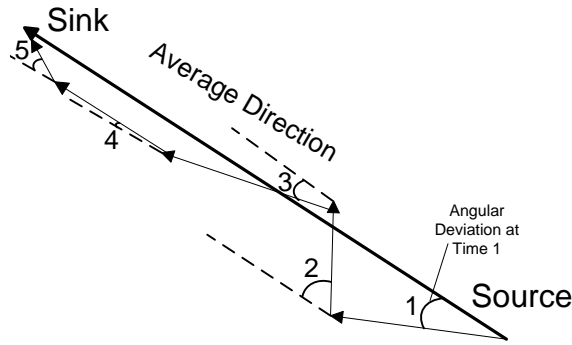


Figure 3.5: Illustration of the method for measuring angular movement versus average direction of movement. Five time steps are shown. All dotted lines are parallel to the average direction of movement.

$\Theta$  is obtained by observing the overall direction of the pedestrian from its source (the point at which it appears) to its sink (the point at which it disappears) as shown in Figures 3.5 and 3.6. In any timestep the pedestrian proceeds exactly the overall direction,  $\Theta$  is 0. In any timestep the pedestrian heads to the left of the overall direction,  $\Theta$  is negative. In any timestep it heads right of the overall direction,  $\Theta$  is positive.  $S$  is obtained by dividing the distance moved over a timestep by the duration of the timestep (1 second).

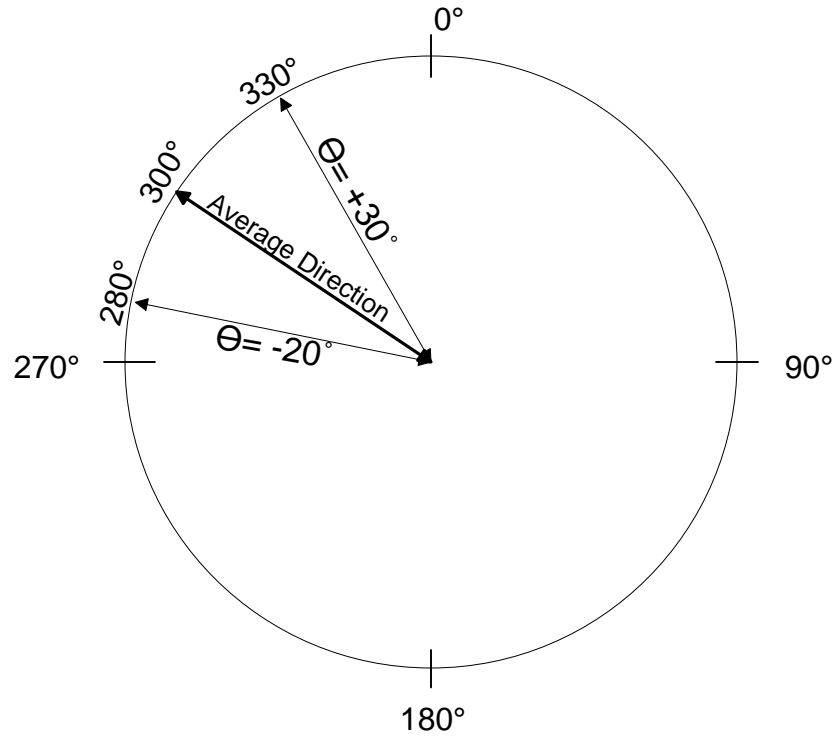


Figure 3.6: Illustration of the method for calculating Angular Deviation,  $\Theta$ . Note that movement left of average movement is defined as negative degrees. Right of average movement is defined as positive degrees.

The two values  $\Theta$  and  $S$  are arranged in a two-element vector  $\mathbf{O}_t$  which represents observation vector  $\mathbf{O}$  at time  $t$ .



## IV. Results and Discussion

This chapter presents and analyzes the results of the experiments described in sections 3.6.1, 3.6.2, and 3.6.3, namely the pilot experiment and the Fryer dataset experiment. Two Multi Observation Continuous Density Hidden Markov Model (MOCDHMM) methods are evaluated: standard MOCDHMM and modified MOCDHMM. The modified method incorporates the solutions to mathematical errors described in Section 3.5.

Section 4.1 of this chapter shows experimental cases of the mathematical errors that arise when applying the standard MOCDHMM algorithm. Section 4.2 details the overall results of the standard MOCDHMM algorithm. Section 4.3 gives the results of the modified MOCDHMM algorithm and makes comparisons between the two algorithms' results to show improvement.

### 4.1 Experimental examples of errors arising from standard MOCDHMM algorithm

This section shows examples of the three errors described in Section 3.4. The examples of this section are not the only occurrences of the described errors across all experiments. This section is included to show that the errors of Section 3.4 arose in experimentation and to help to illustrate what those errors looked like.

#### 4.1.1 Example of error 1: badly scaled covariance matrix.

An example of a badly scaled covariance matrix during Segmental K Means Training (Section 3.4.1) occurs in the pilot experiment (Section 3.6.1) during training for agent HblendI2. The covariance matrix  $\Sigma_{\text{Error1}}$  shown in Eq. (4.1) is obtained in early Segmental K Means training iterations.

$$\Sigma_{\text{Error1}} = \begin{bmatrix} 0.316446 & -0.035390 \\ -0.035390 & 0.003958 \end{bmatrix} \quad (4.1)$$

The matrix determinant of  $\Sigma_{\text{Error1}}$  is on the order of  $10^{-17}$ . Because this matrix is so badly scaled, small changes in the way matrix implementation rounds the values caused large changes in the inverted value or rendered it not invertible. Large changes in the inverted value of an covariance matrix cause large changes in the log-likelihood calculated for classification; such large changes can negatively impact classification accuracy. A non-invertible matrix stops the training procedure and prevents a classification experiment.

#### ***4.1.2 Example of error 2: division by zero in alpha scaling.***

This section demonstrates error 2, division by zero in alpha scaling. Division by zero in alpha scaling is an uncommon problem for discrete density HMMs. It is unlikely that some observation symbol will have zero probability of being produced in all of the hidden states. However, for continuous-density HMMs, error 2 is not at all uncommon.

We demonstrate error 2 in two parts. First, we demonstrate an example in which scaling is required in the Forward Backward algorithm (Section 3.3.4). Second, we show a different example in which scaling causes a division by zero error. Both examples are drawn from the pilot experiment described in Section 3.6.1.

#### ***Example showing that scaling is needed in the Forward Backward algorithm.***

Table 4.1 demonstrates how fast the *unscaled* calculation of  $f(\mathcal{S}|\lambda)$  can cause underflow in a standard IEEE 754 machine using double precision. Table 4.1 shows the calculation of the likelihood  $f(\mathcal{S}_1^D|\lambda_H)$  of realizing the first sequence of agent D given HMM H. Note that the calculation underflows by timestep six. In actuality the first sequence of the agent D dataset is 50 timesteps long.

The scaling techniques given in Section 3.3.4 preclude this underflow, but cause the division by zero error of the next paragraph.

***Example showing scaling can produce division by zero errors.*** Let  $\mathbf{w}_t = [6.065 \quad -2.339]$ . The two-element vector  $\mathbf{w}_t$  is the  $t = 17$  observation from  $\mathcal{S}_1^D$ , the first sequence of agent D data. Consider the likelihood  $\mathcal{L}(\mathcal{S}_1^D|\lambda_H)$  of having produced this observation

Table 4.1: Table showing the *unscaled* forward variable matrix for calculation of  $\alpha_t(i)$ . The calculation shown here is for the likelihood  $f(\mathcal{S}_1^D|\lambda_H)$  of the first agent D training sequence given the HMM trained for agent H. Rows show timesteps and columns show states. Note that the calculation underflows by timestep 7.

timestep( $t$ )/state( $i$ )	0	1
1	1.11636217484e-10	4.14935147103e-10
2	1.03581807032e-34	2.68516010555e-36
3	5.07636051352e-186	1.89071530281e-187
4	1.04801759845e-200	8.06208632366e-201
5	5.1847394146e-217	3.47436325055e-217
6	3.49735376633e-222	4.4810454662e-221
7	0.0	0.0
8	0.0	0.0
$\vdots$	$\vdots$	$\vdots$
50	0.0	0.0

given HMM H. For reference, the trained HMM H is shown in Figure 4.1. To calculate the likelihood, we need to compute the likelihoods  $b_0(\mathbf{w}_t)$  and  $b_1(\mathbf{w}_t)$  of realizing  $\mathbf{w}_t$  given that the process is in hidden states 0 and 1 respectively.

The formula for the log-likelihood of the multivariate Gaussian is in Eqs. (4.2) and (4.3), in which  $k = 2$  is the dimensionality of the observation vectors,  $b_i(\mathbf{w}_t)$  is the likelihood of realizing observation  $\mathbf{w}_t$  at hidden state  $i$ ;  $\mu_i$  and  $\Sigma_i$  are the mean vector and covariance matrix respectively for the PDF at hidden state  $i$ .

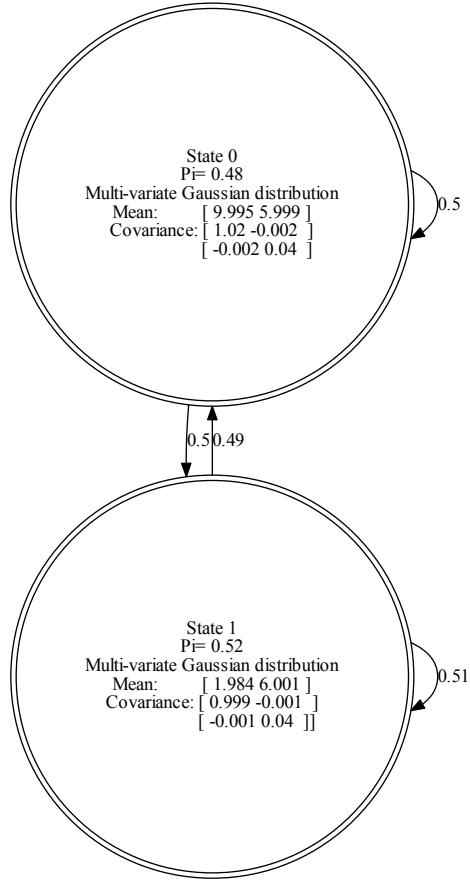


Figure 4.1: The HMM produced by K Means Learning for agent H.

$$\ln b_0(\mathbf{w}_t) = -\frac{k}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma_0| - \frac{1}{2} (\mathbf{w} - \mu_0) \Sigma_0^{-1} (\mathbf{w} - \mu_0)^T \quad (4.2)$$

$$\ln b_1(\mathbf{w}_t) = -\frac{k}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma_1| - \frac{1}{2} (\mathbf{w} - \mu_1) \Sigma_1^{-1} (\mathbf{w} - \mu_1)^T \quad (4.3)$$

Eqs. (4.2) and (4.3) permitted us to calculate the log-likelihood  $\ln b_0(\mathbf{w}_t)$  and  $\ln b_1(\mathbf{w}_t)$  of observation  $\mathbf{w}_t$ , given in Eqs. (4.4) and (4.5). However, the Forward Backward Algorithm uses *likelihood* not log-likelihood, which is a much smaller value.

$$\ln b_0(\mathbf{w}_t) = -878.53 \quad (4.4)$$

$$\ln b_1(\mathbf{w}_t) = -877.18 \quad (4.5)$$

The *likelihood* values  $b_0(\mathbf{w}_t)$  and  $b_1(\mathbf{w}_t)$  significantly underflowed our standard IEEE 754 machine, whose smallest floating-point representation was  $1 \times 10^{-499}$ . Thus, the sum of the intermediate forward variables for all hidden states  $\sum_{i=1}^N \omega_t(i)$  was calculated to be 0 and the calculation of the scaling factor  $c_t$  was undefined. In our experiment, an undefined  $c_t$  value caused *not a number (nan)* results for the log-likelihood  $\mathcal{L}(\mathcal{S}|\lambda)$  of a sequence given an HMM. Such *nan* results caused entire sequence sets to be classified incorrectly.

#### 4.1.3 Example of error 3: biased classifier.

This section shows an example of error 3 of Section 3.4.3, the biased classifier. Table 4.10, located in Section 4.3.3 illustrates the biased classifier problem in Section 3.4.3. Table 4.10 shows a *mean log-likelihood matrix*; the mean log-likelihoods are shown for the sequence sets  $\mathcal{E}^{\text{ROW}}$  given the HMMs  $\lambda_{\text{COLUMN}}$ . For more information on mean log-likelihood see Eq. (4.6). Consider mean log-likelihoods for the SAFE HMM. The SAFE HMM recognizes the SAFE sequence set far better than it does the CARELESS sequence set because of the higher log-likelihood for the SAFE sequence set (-229 versus -270). Likewise, the CARELESS HMM recognizes the CARELESS sequence set far better than it does the SAFE sequence set (-99 versus -118). However, the classifier made up of the CARELESS and SAFE HMMs is biased toward CARELESS since most SAFE sequences will be classified CARELESS (-118 is much greater than -229). This leads to high accuracy rates when classifying CARELESS sequences and low accuracies when classifying SAFE sequences.

## 4.2 Results of experiment with standard MOCDHMM algorithm

The results of experiments described in this section use the standard MOCDHMM algorithm described in Section 3.3. The two experiments presented are the pilot experiment and Fryer experiment described in Sections 3.6.1 and 3.6.3. The results in this section provide a baseline against which to compare the results of the modified MOCDHMM algorithm. This sections shows that the standard algorithm fails and produces erratic behavior. Furthermore, this section define a method of quantifying how erratic the performance of the algorithm is by defining a concept of a failed test case versus a successful one.

### 4.2.1 Unblended pilot experiment results: Standard MOCDHMM.

The results of the unblended pilot experiment are in Table 4.2. In Table 4.2, one entry represents the equal weighted accuracy (EWA) when the **row** agent sequence set is classified against the **column** agent sequence set. In more technical terms, Table 4.2 gives the accuracy of the classifier  $\Lambda_{\text{ROW}, \text{COLUMN}}$  composed of HMMs  $\lambda_{\text{ROW}}$  and  $\lambda_{\text{COLUMN}}$  when classifying sequence sets  $\mathcal{E}^{\text{ROW}}$  and  $\mathcal{E}^{\text{COLUMN}}$ . Table 4.2 shows that the standard MOCDHMM algorithm properly separates agents described in Section 3.6.1. Most accuracies are approximately 100%. The exceptions are experiments for D vs. H and G vs. I, which are both approximately 50%. Both D vs. H and G vs. I contain cases in which log-likelihoods evaluate to *not a number*(*nan*). Such *nan* results show that division by zero (error 2) described in Section 3.4.2 may have occurred.

### 4.2.2 Blended pilot experiment results: Standard MOCDHMM.

This section gives the performance results of the blended pilot experiment under the standard MOCDHMM. The section then gives a brief characterization of the error resistance of standard MOCDHMMs in the blended pilot experiment. (The error resistance does not change in modified MOCDHMMs.)

Table 4.2: Results of unblended pilot experiment for standard MOCDHMM: EWA accuracy

		Sequence Set 2								
		A	B	C	D	E	F	G	H	I
Sequence Set 1	A	1.0	0.985	1.0	0.9825	0.980	0.980	1.0	1.0	
	B		1.0	0.980	1.0	1.0	1.0	1.0	0.980	
	C			1.0	0.980	1.0	1.0	1.0	1.0	
	D				1.0	1.0	1.0	0.54	1.0	
	E					0.985	1.0	1.0	1.0	
	F						1.0	1.0	1.0	
	G							1.0	0.50	
	H									1.0
	I									

#### 4.2.2.1 Performance of blended pilot experiment: Standard MOCDHMM.

Recall the agent types, labeled A through I, which were defined in Table 3.2. Figure 4.2 shows a series of plots reporting the results for the blended pilot experiment under the standard MOCDHMM. Figure 4.2 is arranged according to the agents to be classified. For example, the intersection of the row “A” with the column “I” represents the test cases in which the algorithm attempts to classify agent A sequences blended with agent I sequences. The x-axis within each plot represents the blend density, described in Section 3.6.2 and illustrated in Figure 3.3. The y-axis within each plot represents the Equal Weighted Accuracy (EWA) of the combined agent classification. According to this scheme, 1.0 represents perfect classification accuracy, 0.5 represents the classification accuracy expected of random guessing, and 0.0 represents no correct classifications at all. Note that the y-axis of each plot ranges from 0.5 to 1.0.

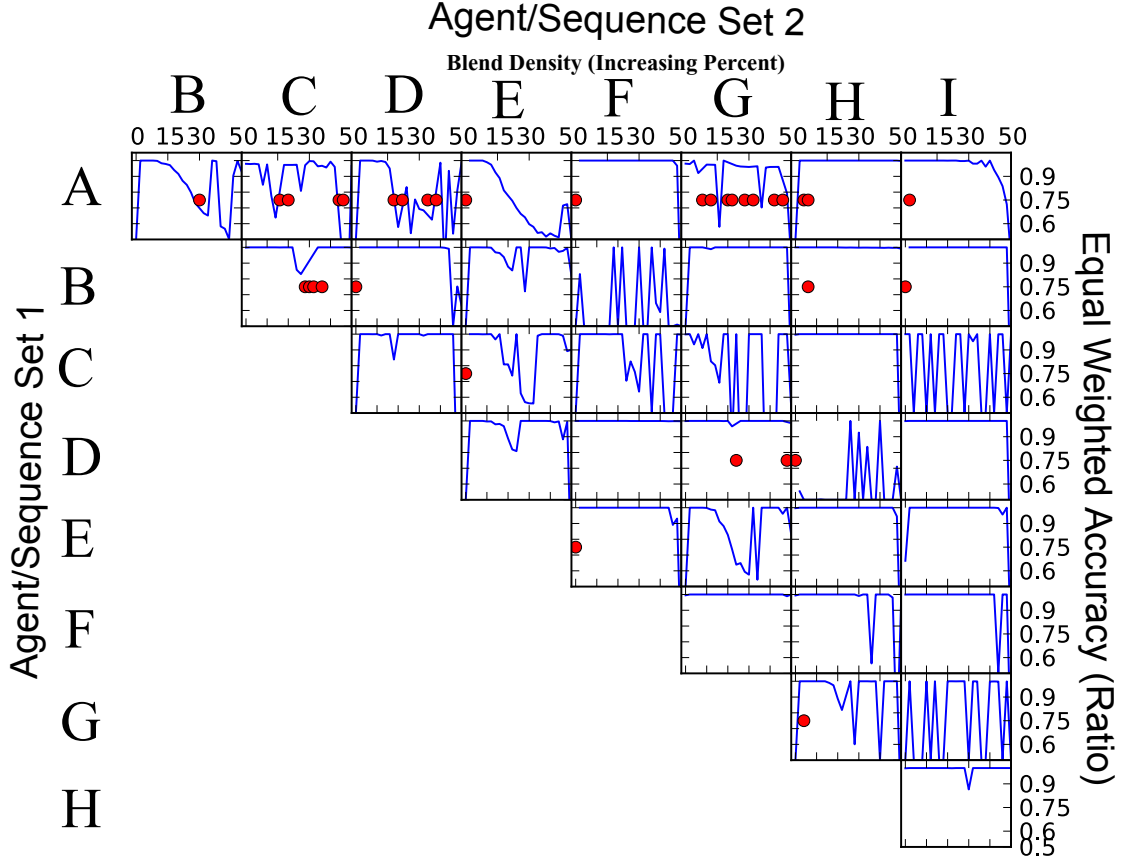


Figure 4.2: Results of the blended pilot experiment under standard MOCDHMM algorithm. Any blend density for which an HMM will not train due to matrix singularity is denoted by a large dot.

In Figure 4.2, any blend density for which an HMM will not train correctly due to the matrix conditioning problem described in Section 3.4.1 is represented by a large dot in the plot. Furthermore, all sequences whose log-likelihood evaluates to *not a number* (*nan*) for either HMM in a classifier are labeled “incorrectly classified”. In other words, the classifier makes an incorrect classification for sequence  $S$  under classifier  $\Lambda_{X,Y} = \{\lambda_X, \lambda_Y\}$  anytime either  $\mathcal{L}(S|\lambda_X)$  or  $\mathcal{L}(S|\lambda_Y)$  compute to *nan*.



We expect the classification accuracy to decrease as the blend density increases since the training data presented to the classifier have more errors at higher blend densities. Note that indeed the accuracy decreases as the blend density increases.

Note further that large dots denote several cases for which HMMs are not able to be trained via Segmental K Means Learning due to the existence of a badly conditioned covariance matrix somewhere along its training process. Because no valid classifier was obtainable for those blend densities, no sequences were tested for that blend density, and no accuracy was obtained. The trend line connects only blend densities for which valid HMMs exist and thus does not indicate an accuracy for cases which have a large dot <sup>1</sup>.

Moreover, note that several pairs of agents exhibited poor performance at low blend densities ( $\leq 6$ ): A vs. B, A vs. D, A vs. H, B vs. E, B vs. F, B vs. G, C vs. D, C vs. F, D vs. E, D vs. H, E vs. I, and G vs. H, and G vs. I. This may seem a surprising result since they are trained using the least confused training data. However, the division by zero error described in Section 3.4.2 provides an explanation. Note that we expect this error more commonly for cases in which sequences from widely different distributions were classified. Division by zero in sequences with widely different distributions is more common because the error occurs when one observation in either sequence evaluates to near-zero likelihood for all hidden states in the HMM for the other sequences. If an HMM has been trained on sequences that are extremely different from the test sequences, low likelihoods are more common. Consider cases of poor performance seen in Figure 4.2; for the classification of two notional sequence sets  $\mathcal{E}^X$  and  $\mathcal{E}^Y$ , in least one sequence set  $\mathcal{E}^X$  predominantly produced *nan* results against the HMM  $\lambda_Y$  trained for the other sequence set  $\mathcal{E}^Y$ . Thus, *nan* results caused half of the cases to be judged “incorrect”, driving accuracies

---

<sup>1</sup>Note that if training data produced such a badly conditioned matrix in a live (operational) system, no classifier could be trained for that data. The system would have no recourse but to raise an exception to a user. Thus, we do not attempt test cases based on data that produces badly conditioned matrices; we annotate such test cases with a large dot

of 0.5. In rare cases, both  $\mathcal{E}^X$  and  $\mathcal{E}^Y$  produced *nan* results against their opposite HMMs, driving accuracies of 0.0.

Lastly, note that there are several experiments that exhibit return to accuracies of 0.5 erratically. For example, the test cases involving agent C and agent I demonstrate performance that alternates between close to 1.0 (very good) and close to 0.5 (close to random guessing). This poor performance is due to mathematical problems with the algorithms described in Section 3.3 (cited from previous works in [41]). We are able to determine that the source of poor classification is due to mathematical error because we know that *nan* is returned as the log-likelihood in those cases. However, we were not able to determine the cause of such errors, which is an area for future work (see Section 5.3.2).

We seek to characterize the performance of the standard MOCDHMM algorithm using a single number to allow comparison with the modified MOCDHMM algorithm. One might expect the best characteristic of performance is the average accuracy across all test cases. However, obtaining average accuracy across trials is not possible because test cases whose classifier training failed (large dots in Fig. 4.2) do not give an accuracy. Thus, to best characterize the results of Figure 4.2, we define all test cases as either *failures* or *successes*. Note that in Fig. 4.2, most test cases are either very close to perfect ( $> 0.95$ ) or very close to the performance of random guessing ( $\approx 0.50$ ). Thus, the cutoff value  $r$  between failure and success can be almost any value in the range  $0.50 < r < 0.95$  with little impact on the number of successes or failures counted. As such,  $r = 0.75$  is selected because it is half way between random guessing and perfect accuracy. Results for the standard MOCDHMM against the pilot dataset are summarized in Table 4.3.

#### 4.2.2.2 *Characterizing the error resistance of MOCDHMMs.*

The pilot experiment whose results are described in Figures 4.2 and 4.3 demonstrates that MOCDHMMs are highly resistant to incorrectly classified sequences in their training data. They match the *error resistant* profile in Figure 3.4. In general, they do not exhibit

Table 4.3: Summary of results for blended pilot experiment: Standard MOCDHMM algorithm. Success is defined as accuracy  $\geq 0.75$ .

Successes	746
Total tested cases	936
Success rate	0.797009

a loss of classification accuracy until high blend densities. This finding is consistent with that of other HMMs; MOCDHMMs and traditional HMMs are robust to errors in training data.

The degree to which MOCDHMMs are resistant to error-prone (blended) data is notable. In most cases, blend densities of 40% or more still produce accuracies close to 1.0 (perfect). The decline in accuracy occurs rapidly, and only in the highest blend densities (close to 50%).

#### 4.2.3 *Fryer dataset experiment results: standard MOCDHMM.*

This section reports the results of the standard MOCDHMM algorithm in classifying the Fryer dataset described in Section 3.6.3. Table 4.4 reports the accuracy of the classifier  $\Lambda_{\text{ROW,COLUMN}}$  using two hidden states and the standard MOCDHMM algorithm. The rows and columns in Table 4.4 both represent agent class names; each entry represents the accuracy for sequence set  $\mathcal{E}^{\text{ROW}}$  when classified using classifier  $\Lambda_{\text{ROW,COLUMN}}$ . For example, the entry in the AWARE row, CARELESS column represents the overall accuracy in **only** the AWARE dataset when classifying AWARE data against CARELESS data.

Results for HMMs of only two hidden states are reported for the standard MOCDHMM algorithm. HMMs with greater than two hidden states failed to train properly under standard MOCDHMM due to the badly conditioned covariance matrix problem described in Section 3.4.1. The overall accuracy of the Fryer standard MOCDHMM experiment is calculated by taking the mean of all of all values in Table 4.4. The mean

accuracy of Table 4.4 is calculated as 0.59324. This mean accuracy is recorded in the standard, two-state entry in Table 4.7; Table 4.7 is a comparison of standard and modified MOCDHMM performance.

Note that Table 4.4 reveals some bias in the classifier. Most accuracy values above 0.5 on one side of the diagonal have corresponding values below 0.5 on the other side of the diagonal. This indicates that the classifier is selecting one class dominantly. This biased classifier effect is addressed mathematically in Sections 3.4.3 and 3.5.3 dealing with scored evaluation.

Table 4.4: Accuracies for individual sequence sets: Fryer dataset experiment using standard algorithm and two states. Rows represent the sequence sets. The classifier for each entry is  $\Lambda_{\text{ROW},\text{COLUMN}}$ .

		HMM 2				
		AWARE	CARELESS	COMBAT	SAFE	STEALTH
Sequence Set $\mathcal{E}$ and HMM 1	AWARE		0.7598	0.6537	0.6702	0.7853
	CARELESS	0.4103		0.3722	0.4183	0.6924
	COMBAT	0.5198	0.7629		0.7709	0.8174
	SAFE	0.3445	0.8080	0.4065		0.8300
	STEALTH	0.4209	0.6752	0.3478	0.3989	

### 4.3 Results of experiments with modified MOCDHMM algorithm

This section will show that the modified MOCDHMM algorithm provides modest but numerically quantifiable improvement over the standard MOCDHMM algorithm. The experiments presented in this section were performed using the modified MOCDHMM algorithm.

#### 4.3.1 Unblended pilot experiment results: Modified MOCDHMM.

Table 4.5 gives the results of the unblended pilot experiment using the modified MOCDHMM algorithm. We expect low blend densities and widely separated distributions to produce *nan* results due to error 2. Accordingly, we hypothesized that all cases of EWA close to 50% in the standard MOCDHMM unblended pilot experiment (Figure 4.2) were due to error 2 and that those cases would thus show 100% accuracy under the modified MOCDHMM algorithm. Figure 4.5 shows that this is not entirely true. The accuracy for D vs. H (54% accuracy with the standard MOCDHMM algorithm) was improved to 100% in this experiment, but the accuracy for G vs. I (50% accuracy with the standard MOCDHMM algorithm) was not improved. We can thus conclude that error 2 is not the cause of *nan* results for G vs. I and that another error causes the problem. Discovering the source of error for G vs. I is an area for future work.

Table 4.5: Results of unblended pilot experiment for modified MOCDHMM: EWA accuracy

		Sequence Set 2								
		A	B	C	D	E	F	G	H	I
Sequence Set 1	A	1.0	0.985	1.0	0.9825	0.980	0.980	1.0	1.0	
	B			1.0	0.980	1.0	1.0	1.0	1.0	0.980
	C				1.0	0.980	1.0	1.0	1.0	1.0
	D					1.0	1.0	1.0	1.0	1.0
	E						0.985	1.0	1.0	1.0
	F							1.0	1.0	1.0
	G								1.0	0.50
	H									1.0
	I									

#### 4.3.2 *Blended pilot experiment results: Modified MOCDHMM.*

Figure 4.3 shows the results of the blended pilot experiment using the modified MOCDHMM algorithm. The results of the blended pilot experiment do not use the scored evaluation fix described in Section 3.5.3 because the classifiers do not exhibit the biased classifier problem (Section 3.4.3) that scored evaluation is intended to fix. The plots in Figure 4.3 are read in the same way as those for the blended pilot experiment standard algorithm shown in Figure 4.2; blend density is listed on the x-axis and accuracy is listed on the y-axis. Note that unlike Figure 4.2 for the standard MOCDHMM, Figure 4.3 has no large dots indicating cases of failed classifiers. This is because all classifiers trained properly; each blend density within each pair of agents has a valid accuracy plotted. This is opposed to the standard MOCDHMM in which 35 classifiers failed to train.

The continued erratic occurrence of poor (0.5) accuracy exhibited by the modified MOCDHMM algorithm is an unexpected result. Note that in both Figures 4.2 and 4.3, test cases in G vs. I, C vs. I, C vs. G, B vs. F, and D vs. H alternate between high accuracy and low accuracy. We expected that fixing the division by zero error described at Section 3.4.2 would prevent the return of *not a number (nan)* results from the forward algorithm. In turn, we expected that the erratic occurrences of poor performance would be precluded. Instead, observe that while the modified MOCDHMM algorithm eliminates cases of poor performance from low blend densities, it does not prevent poor performance in higher blend densities. Of the 190 failures realized by the standard MOCDHMM algorithm, the modified MOCDHMM algorithm corrects 42 and fails to correct 148. There are no cases of success under the standard MOCDHMM algorithm that were failures under the modified MOCDHMM algorithm. The failure cases under the modified MOCDHMM algorithm are due to *nan* log-likelihoods. We have not been able to determine what caused the *nan* log-likelihoods; such a determination is an area for future work (Section 5.3).

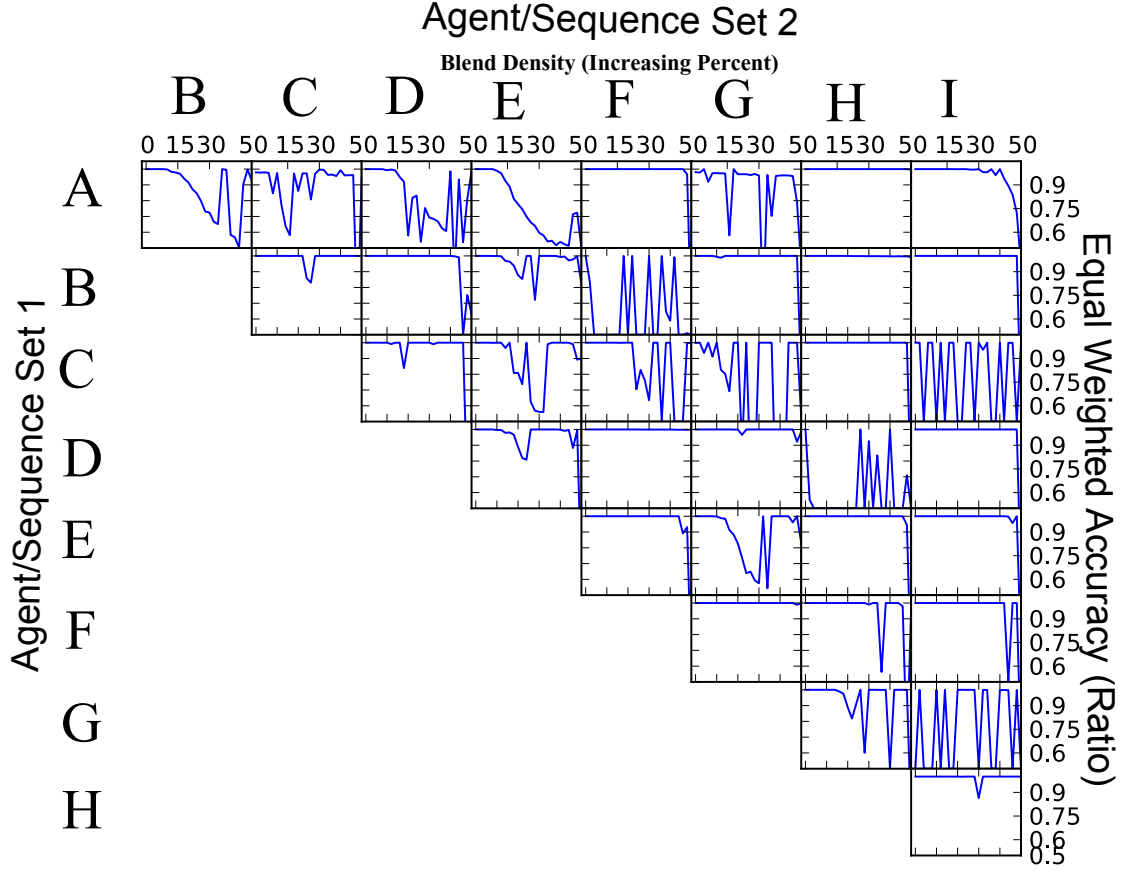


Figure 4.3: Results of the blended pilot experiment using the modified MOC DHMM.

Table 4.6 summarizes the results in terms of separating test cases into *successes* and *failures* as defined in Section 4.2.2. Note that the total success rate of the modified algorithm reported in Table 4.6 is about 84% while the total success rate of the standard algorithm, reported in Table 4.3 is about 79%, suggesting an approximate 5% improvement using the modified method. In order to verify the 5% improvement is statistically significant, the two-sample test for equality of proportions is used. The two-sample test for equality of proportions determines the probability (p-value) that the difference in two samples assumed to be from the same population occurred due to random chance. The p-value for the standard MOC DHMM algorithm and modified MOC DHMM algorithm in

this experiment was about 0.01. We thus conclude at 95% confidence that the modified MOCDHMM algorithm produced an actual improvement in success rate. The increase in the success rate owed to elimination of untrainable classifiers (error 1) and correction of test cases at low blend densities (error 2).

Table 4.6: Summary of results from the blended pilot experiment using the modified MOCDHMM. Success is defined as accuracy  $\geq 0.75$ .

Successes	788
Total tested cases	936
Success rate	0.841880

#### 4.3.3 *Fryer dataset experiment results: modified MOCDHMM.*

This section details the results of the Fryer experiment described in Section 3.6.3 using two of the fixes for the modified MOCDHMM described in Section 3.5. The end of the section contains a brief analysis to show statistically significant the performance difference between the standard MOCDHMM algorithm and the modified MOCDHMM algorithm.

The experiment was conducted eight times as follows. Four cases used fix 1 (pseudoinverse in Segmental K Means Training) and fix 3 (scored evaluation) in conjunction; these cases are referred to here as the *modified-scored* cases. Four cases used fix 1 WITHOUT fix 3 (unscored evaluation); these cases are referred to here as the *modified-unscored* cases. Both the modified-scored and modified-unscored cases were tested with two, three, four, and five hidden states, resulting in eight total cases. (Although we ran experiments with six and seven hidden state HMMs, we found that those HMMs contained null states. HMMs of greater than five states thus produced poor accuracy and their results are not given here.) Division by zero errors were not encountered in the Fryer experiment; thus fix 2 has no effect on (and was not used in) the Fryer experiment.



The overall accuracies in Table 4.7 report the mean accuracy results for those eight cases. Table 4.7 includes one result of the standard MOCDHMM algorithm incorporating no fixes. The standard MOCDHMM result is included for the sake of comparison to the modified MOCDHMM results. Note that the standard MOCDHMM cannot produce valid classifiers for any number of hidden states greater than two in this experiment. Accordingly, the corresponding entries in Table 4.7 are populated with “N/A”. Note also that the two-state standard MOCDHMM experiment and the two-state modified-unscored MOCDHMM experiment are in effect the same experiment and produce the exact same result for these reasons: the pseudoinverse fix was not required for proper two-state training and the division by zero error did not occur in the Fryer experiment. Thus, for two states, the only difference between modified and unmodified MOCDHMM algorithms is scoring.

Table 4.7: Accuracies for the Fryer Experiment using the standard algorithm, the modified-unscored algorithm, and the modified-scored algorithm. Note that the standard algorithm would fail to train an HMM for greater than two hidden states due to the badly conditioned covariance matrix problem.

	Standard	Modified-Unscored	Modified-Scored
2 States	0.5932	0.5932	0.5209
3 States	N/A	0.6278	0.5996
4 States	N/A	0.6283	0.6124
5 States	N/A	0.5882	0.6119

The entries in Table 4.7 are mean accuracies across each pairwise experiment (AWARE vs. CARELESS, CARELESS vs. COMABT, etc...). To illustrate the meaning of one entry in Table 4.7, we include Table 4.8. Table 4.8 is read exactly as Table 4.4; e.g. the entry for AWARE and CARELESS corresponds to the accuracy for only AWARE

sequences when the modified-unscored algorithm is used to classify AWARE sequences versus CARELESS sequences. The average accuracy of Table 4.8 is recorded under the entry for modified-unscored and five states in Table 4.7.

Table 4.8: Accuracies for individual sequence sets: Fryer dataset experiment using modified-unscored algorithm and five states. Rows represent the sequence sets for which accuracy is given when the classifier is composed of  $\lambda_{\text{ROW}}$  and  $\lambda_{\text{COLUMN}}$ .

		HMM 2				
		AWARE	CARELESS	COMBAT	SAFE	STEALTH
Sequence Set $\mathcal{E}$ and HMM 1	AWARE		0.6602	0.3844	0.9990	0.6441
	CARELESS	0.7791		0.6253	0.9965	0.7841
	COMBAT	0.7089	0.7199		0.9815	0.697
	SAFE	0.0020	0.0115	0.0225		0.0030
	STEALTH	0.6021	0.6997	0.4494	0.9935	

Several important results are observable from Table 4.7. First, note that the accuracy in every case is above random guessing of 50%. Above random results are particularly important in this experiment since the amount of information is limited in the datasets provided to the classifier. Above random performance suggests it is possible to make inferences on the class of entities in full motion video using their tracks alone. Furthermore, Table 4.7 demonstrates that modifying the original MOCDHMM algorithm to permit more than two states does allow for a small increase in performance. Note that performance increased for both the modified-scored and modified-unscored case from two to three states. Note further that scoring does increase accuracy in the five state case.

Table 4.7 also contains a few unexpected results. In the modified-scored results note a steady increase in classifier performance, but note the modified-unscored case reveals

a drop in performance in the five state model. Furthermore, note that the overall best performance was in an unscored case, that for four hidden states. In the two hidden state case, the modified-scored algorithm performs significantly worse than the modified-unscored algorithm (although it still outperforms random guessing). Yet we also find that the scored algorithm outperforms the unscored in the five hidden state case.

To help explain why scoring helps most in the five state models, we include Tables 4.9 and 4.10. We term Tables 4.9 and 4.10 *mean log-likelihood matrices*, which list the mean log-likelihood  $\mathcal{L}_\mu(\mathcal{E}^{\text{ROW}}|\lambda_{\text{COLUMN}})$  of the sequence sets  $\mathcal{E}^{\text{ROW}}$  represented by the rows against HMMs  $\lambda_{\text{COLUMN}}$  represented by the columns. For example, the entry in the AWARE row and the CARELESS column represents the mean log-likelihood value when the AWARE dataset is evaluated against the HMM trained on CARELESS data. Table 4.9 gives log-likelihoods for HMMs of two hidden states, and Table 4.10 gives log-likelihoods for HMMs of five hidden states. Mathematically, we obtain the mean log-likelihood value using Eq. (4.6), where  $\mathcal{L}_\mu$  is the mean log-likelihood which makes up the entries of Table 4.9,  $\mathcal{E}^{\text{ROW}}$  is the sequence set of the class ROW,  $\mathcal{S}_l^{\text{ROW}}$  is a particular sequence of class ROW,  $L^{\text{ROW}}$  is the number of sequences in  $\mathcal{E}^{\text{ROW}}$ , and  $\lambda_{\text{COLUMN}}$  is the HMM trained on class COLUMN training data.

$$\mathcal{L}_\mu(\mathcal{E}^{\text{ROW}}|\lambda_{\text{COLUMN}}) = \frac{\sum_{l=1}^{L^{\text{ROW}}} \mathcal{L}(\mathcal{S}_l^{\text{ROW}}|\lambda_{\text{COLUMN}})}{L^{\text{ROW}}} \quad (4.6)$$

The ideal case for Tables 4.9 and 4.10 occurs when the highest value in every row occurs on the diagonal. In this case, a sequence set *is recognized by* (per Section 3.4.3 definition) its own HMM better than any other HMM. Note that in Table 4.9 (2 state case), the highest value *is* on the diagonal in all rows except for SAFE; in the SAFE row it is very close (-133.04 is on the diagonal while -132.50 is the highest value in the row). We do not expect that scoring will help much in this case since scoring only recenters the

Table 4.9: Mean log-likelihood matrix: mean log likelihoods versus trained HMMs for two hidden states. Rows represent sequence sets and columns represent HMMs.

		HMM $\lambda$				
		AWARE	CARELESS	COMBAT	SAFE	STEALTH
Sequence Set $\mathcal{E}$	AWARE	-132.45	-141.36	-136.41	-133.42	-141.66
	CARELESS	-157.61	-154.20	-156.58	-156.83	-160.46
	COMBAT	-152.92	-155.12	-149.78	-152.35	-153.92
	SAFE	-132.50	-140.54	-136.54	-133.04	-141.69
	STEALTH	-207.73	-207.08	-204.23	-207.43	-201.55

Table 4.10: Mean log-likelihood matrix: mean log likelihoods versus trained HMMs for five hidden states. Rows represent sequence sets and columns represent HMMs.

		HMM $\lambda$				
		AWARE	CARELESS	COMBAT	SAFE	STEALTH
Sequence Set $\mathcal{E}$	AWARE	-94.14	-111.75	-96.91	-228.01	-97.28
	CARELESS	-119.94	-99.17	-101.43	-269.97	-124.31
	COMBAT	-107.36	-124.11	-101.27	-257.70	-109.02
	SAFE	-96.85	-118.85	-100.41	-229.17	-99.77
	STEALTH	-153.96	-206.52	-157.26	-348.94	-105.44

classification based on mean log-likelihood score. It may be that most classification errors in the two-state experiment are due to mean log-likelihood scores being close together rather than due to biased classifiers. However, adding hidden states to the HMMs provides some log-likelihood separation.

In the five-state experiment (Table 4.10) the difference between the diagonal values and the nearest other values on the same rows is greater than in the two-state experiment (Table 4.9). Having a diagonal value much greater than the other values in a row aids in classification accuracy because there is more separation on which to classify. Moreover, the five-state log-likelihoods are similar to the two-state log-likelihoods in that the value on the diagonal is the highest in each row except for SAFE; having highest log-likelihood values on the diagonal aids in classification accuracy because we can expect to classify each sequence correctly on average.

By contrast to the two-state experiment, the five-state experiment (Table 4.10) represents an experiment in which scored evaluation may help. The SAFE vs. SAFE value (-229.17) in the 4th row of Table 4.10 is very low compared with the other values on this row. We thus expect SAFE data to be consistently classified incorrectly as a different class. Table 4.8 confirms the expectation of low SAFE accuracy.

In Table 4.8, accuracies for the five-state modified-unscored test are broken down into those obtained for the individual sequence sets. Each entry represents a single experiment in which sequence set  $\mathcal{E}^{\text{ROW}}$  is classified against  $\mathcal{E}^{\text{COLUMN}}$  using the classifier  $\Lambda_{\text{ROW},\text{COLUMN}} = \{\lambda_{\text{ROW}}, \lambda_{\text{COLUMN}}\}$ . The accuracy given is that for classifying **only**  $\mathcal{E}^{\text{ROW}}$ . (To find the corresponding accuracy for  $\mathcal{E}^{\text{COLUMN}}$ , look on the opposite side of the diagonal where the labels are reversed.) Note that across the row for SAFE, the accuracies are near zero, whereas in the column for SAFE the accuracies are near perfect; this means that sequences in  $\mathcal{E}^{\text{SAFE}}$  are almost always classified incorrectly. Low SAFE accuracy is due to the low mean log-likelihoods  $\mathcal{L}_{\mu}(\mathcal{E}^{\text{SAFE}}|\lambda)$  given in Table 4.10.

However, that the SAFE HMM *recognizes* SAFE data as well as any other class makes scoring effective for the five-state experiment. A class  $C$  is *recognized* by its own HMM  $\lambda_C$  better than other classes are if the mean log-likelihood  $\mathcal{L}_{\mu}(\mathcal{E}^C|\lambda_C)$  is the highest for **row**  $C$ . An HMM  $\lambda_C$  *recognizes* its own class  $C$  better than other HMMs do if the mean log-

likelihood  $\mathcal{L}_\mu(\mathcal{E}^C|\lambda_C)$  is the highest for **column C**. Although SAFE data *is recognized* by other HMMs better than  $\lambda_{\text{SAFE}}$ , SAFE HMMs *recognize* their own data almost as well as all others (see Section 3.4.3 for definitions of *recognition*). Thus, observe that the SAFE vs. SAFE value in Table 4.10 (-229.17) is almost equal to the highest value (-228.01) in the SAFE column. Scoring then helps by scoring SAFE data close to zero on average and other data less than zero on average. As a result, there is a classification accuracy increase for the 5 state case from unscored to scored evaluation, as seen in Table 4.7.

The analysis above (and that in Chapter 5) assume that the differences of algorithm accuracy in Table 4.7 do not occur by random chance. Thus, the analyses assume differences in Table 4.7 are due to the effects of changing the algorithms. A statistical analysis is now presented to show the differences in mean accuracy are in fact statistically significant.

#### ***4.3.3.1 Remarks on statistical significance of modified MOCDHMM Fryer dataset results.***

The two-sample test for equality of proportions is applied to determine whether the differences in mean accuracy shown in Table 4.7 are statistically significant, that is, that differences in mean accuracy occur due to changes in the algorithm and not due to random chance. The two sample test for equality of proportions is performed at 95% confidence on each pair of mean accuracies in Table 4.7; the sample size used for the test was 9985 (1997 sequences per sequence set times five sequence sets).

The two-sample test for equality of proportions reports that all mean accuracies in Table 4.7 are different from one another with four exceptions. Those exceptions are:

1. The standard MOCDHMM algorithm accuracy and the two-state modified-unscored MOCDHMM algorithm accuracy are not different. Such is expected since those are the same algorithm by two different names.

2. The two-state modified-unscored MOCDHMM algorithm accuracy and the three-state modified-scored MOCDHMM algorithm accuracy are not different. However, intuition suggests that the algorithms do in fact produce different results for two reasons: 1) there is a statistically significant difference between accuracy for two-state and three-state modified-unscored, and 2) there is statistically significant difference between accuracy for scored and unscored three-state modified.
3. There is no statistically significant difference among accuracy for the three-state or four-state modified-scored MOCDHMM algorithm. However, the improvement in accuracy for the modified-scored algorithms for the other numbers of states suggests that increasing the number of hidden states in general increases mean log-likelihood separation among the classes and thus increases scored classifier accuracy.
4. There is no statistically significant difference among accuracy for the two-state, three-state, or four-state modified-unscored MOCDHMM algorithms. It may be that these three algorithms do not in fact produce different performance from one another. This result may support a conclusion that increasing the number of hidden states increases mean log-likelihood difference among the classes but causes classifier bias. Increased mean log-likelihood separation among the classes tends to increase classifier accuracy while classifier bias (in an unscored algorithm) tends to decrease classifier accuracy. As a result, increasing the number of hidden states would produce negligible effect on accuracy for an unscored classifier.

#### ***4.3.3.2 Comparison of modified MOCDHMM algorithm to Mean Euclidean Distance (MED) classification for the Fryer dataset.***

This section gives a comparison of the classification accuracy results Fryer [15] obtained to those obtained by the modified MOCDHMM algorithm. This section provides

Table 4.11: Overall improvement of modified MOCDHMM algorithm over standard MOCDHMM algorithm.

	Pilot experiment	Fryer dataset experiment	
		WITHOUT Scoring	WITH SCORING
Improvement	5%	3.5%	2%
p-value	0.01	3.9E-7	0.006
Sample size	936	9985	9985

a brief overview and the results of the Fryer technique, and then presents a comparison of the Fryer results to those of the modified MOCDHMM algorithm.

Fryer [15] designed a classifier which used K Means Clustering and Mean Euclidean Distance (MED) to classify agents of the same five types classified in this thesis. To use the MED classifier, Fryer preprocessed datasets into vectors of 48 elements. One such 48-vector was obtained for each agent to be classified. The 48-vector contained (among other values) the number of eastward, southward, northward, and westward movements of the agent; that is, the number of sampled positions over the agent's walk that appeared eastward (southward, etc.) of the last sampled position. Fryer used K Means Clustering on training data to obtain a mean vector for each of the five agent types. Then, the MED classifier measured the Euclidean distance from a novel sequence  $S$  to each of the five mean vectors; the MED classifier classified  $S$  according to which mean vector produced the lowest distance. Note that the MED classifier technique ignores the time-series aspect of the data.

Fryer performed two experiments to examine the MED classifier. The first experiment was a pairwise classification in which each sequence to be classified could be one of only two possible agent types. The second experiment was a *five-at-a-time* classification in which each sequence to be classified could be one of any of the five agent types.



When classifying according to the MED on all 48 elements of the vectors, pairwise classification was near perfect. The only incorrect classifications occurred when classifying CARELESS versus SAFE agents; nonetheless the CARELESS versus SAFE trial still scored an 80% accuracy. The overall accuracy of the pairwise experiment was 98%. The results of Fryer’s five-at-a-time experiment were less promising by comparison to his pairwise experiment results. The five-at-a-time experiment obtained accuracies above those expected of a randomly-guessing classifier. The overall accuracy of the five-at-a-time experiment was 32%. The reason for the substantial drop from pairwise to five-at-a-time classification is alluded to in Fryer’s thesis. Fryer states that the dataset studied in the five-at-a-time experiment was “less biased toward specific feature space attributes that were easily separable between classes” [15]. The dataset studied in the five-at-a-time experiment was what we refer to as the Fryer dataset in this thesis. The details of why the pairwise experiment dataset was so easily separable based on the 48-vector discussed above are unclear, as is the meaning of the term *biased* in Fryer’s explanation.

The research of this thesis includes only a pairwise classification experiment. The algorithm which produced the best classification results on the Fryer dataset was the unscored-modified MOCDHMM algorithm, which obtained an accuracy of 63%. Accuracy of 63% is far below that obtained by Fryer in his pairwise experiment. However, such a difference does not indicate that MED classification is more appropriate than MOCDHMMs for this domain. The Fryer pairwise results were obtained on a different dataset than we used in our research, a dataset that was “biased” in some way to be easily separable by the Fryer preprocessing technique [15]. No direct comparison between MED classification and MOCDHMM classification is thus possible given available experimental data. The research of this thesis did not obtain five-at-a-time classification results. Such is an area for future work.

## V. Conclusions and Future Work

This research applied what we term the Multi Observation Continuous Density Hidden Markov Model (MOCDHMM) to classifying entities in Automated Visual Surveillance (AVS). We discovered fundamental mathematical errors in the existing MOCDHMM theory in [41], introduced fixes to counter those errors, and tested those fixes using three datasets.

This chapter offers our conclusions on the research of this thesis. First, the chapter summarizes our main contributions and findings. Next, the chapter comments on the advantages and limitations of the MOCDHMM vice simpler HMMs. Finally, the chapter discusses areas for future work that may render MOCDHMMs more accessible to the research community.

### 5.1 Summary of findings

This section summarizes the main contributions and findings of this research. This section has two subsections. The first subsection gives our analytical contributions to the application of Hidden Markov Models (HMMs) for classification. The second subsection gives findings based upon the empirical results of Chapter 4.

#### 5.1.1 *Analytical contributions.*

This section gives the analytical contributions of this research to HMMs and AVS. This section contains methods introduced in this thesis which are novel to the field. The contributions of this section are supported by mathematical derivation rather than by experimental result. (Findings of experimental results are treated in in Section 5.1.2).

- The Moore Penrose pseudoinverse is applied (Section 3.5.1) to ensure HMM training procedures are valid for Gaussian distributions with full (non-diagonalized)

covariance matrices. Previous work ensured covariance matrices would invert by diagonalizing them, thus losing potentially valuable covariance information.

- The Scaled Forward Backward algorithm is modified (modification in Section 3.5.2) to prevent a division by zero error (error 2) when determining the likelihood  $\mathcal{L}(\mathcal{S}^A|\lambda_B)$  of realizing sequence  $\mathcal{S}^A$  from class  $A$  given an HMM  $\lambda$  which was trained for another class  $B$ . Error 2 occurs only in widely separated classes and is not discussed in any literature we found.
- The concepts of *HMM recognition* are defined. We use recognition to mathematically describe the *biased classifier*  $\Lambda_{AB}$  which predominantly selects class  $A$  when it should select class  $B$  (Section 3.5.3).
- Biased classifiers are analyzed by introducing *mean log-likelihood matrices*, whose columns reveal vulnerability to bias.

### 5.1.2 Empirical findings.

This section gives the conclusions supported by the experimental results in Chapter 4. Where applicable, it refers to the appropriate table or section to support the conclusion.

- The modified MOCDHMM algorithm produces statistically significant improvement in classification accuracy for the pilot experiment, the Fryer dataset experiment without scoring, and the Fryer dataset experiment with scoring. Table 4.11 gives the improvement and the p-value probability that the improvement is due only to sampling error when using the two-sample test for equality of proportions. The low p-values permit us to conclude with 95% confidence that the improvement is statistically significant and not due to sampling error alone.
- Increasing number of hidden states produces greater separation in mean log-likelihoods (Tables 4.9 and 4.10). However, increasing number of hidden states

may introduce the biased classifier problem (Section 3.5.3). Separation in mean log-likelihoods tends to increase classifier performance while classifier bias tends to decrease classifier performance for unscored MOCDHMM algorithms. Scoring techniques in MOCDHMM algorithms offset some accuracy decrease due to classifier bias; as a result, scored MOCDHMM algorithm accuracy increases as number of hidden states increases to a point (Table 4.7).

- Improvement for the modified MOCDHMM over the standard MOCDHMM demonstrates that fixes for error 1 and error 2 (Section 3.4) are effective in classifying a dataset designed to be easy to separate. Of 190 failures under the standard MOCDHMM algorithm, 42 were improved to successes under the modified MOCDHMM algorithm.
- Above random performance in the Fryer experiment suggests it is possible to make inferences on the class of entities in full motion video (FMV) using their tracks alone (Table 4.7).
- Applying the Moore Penrose pseudoinverse (fix 1) may enable HMMs to be trained with more hidden states than training methods for Gaussians given in literature [41] permit.
- Occurrence of *not a number*(*nan*) results for the log-likelihood  $\mathcal{L}(\mathcal{S}|\lambda)$  in test cases from the modified MOCDHMM blended pilot experiment (Table 4.6) shows that there are errors in classifying by MOCDHMM besides those addressed in fix 1 and fix 2 (Section 3.5).

## 5.2 Advantages and limitations of MOCDHMMs

MOCDHMMs enjoy several advantages over simpler HMMs. MOCDHMMs preserve more information in a training dataset than do simpler HMMs common in present

literature. Preserving information in a training dataset is important because any information lost in preprocessing might be information vital to class separation. MOCDHMMs preserve such information better than simpler HMMs in two ways. First, MOCDHMMs require no quantization and no discretization unlike HMMs elsewhere in literature. Second, MOCDHMMs (as we use them) preserve correlation among the observations in an observation vector by permitting output Probability Density Functions (PDFs) which contain full (non-diagonalized) covariance matrices. By contrast, most HMMs which permit multiple observations and continuous vectors ignore correlation among the observations (see Section 2.3.2).

However, despite the inherent power of MOCDHMMs, they do possess the disadvantages of mathematical complexity and limited scalability. MOCDHMMs are mathematically complex; they produce calculation errors or underflows in a variety of exceptional cases. This thesis presented three failure modes to which MOCDHMMs (but not usually other HMMs) are subject, and proposed fixes for each. However, that this research was unable to diagnose the cause of all errors discovered implies the high complexity involved in transitioning from simple HMMs to MOCDHMMs. Moreover, many datasets will quantize in some meaningful way and do not exhibit covariance across random variables in their observation vectors. For datasets that can be thus quantized, increased mathematical complexity makes MOCDHMMs a poor choice. (Note that we use the term *mathematical complexity* to distinguish from *computational complexity*. MOCDHMMs do not require any steps in training or evaluation that are not required of simple HMMs. Thus, algorithms using a MOCDHMM and a simple HMM complete in like computational time.)

In addition to the disadvantage of mathematical complexity, MOCDHMMs possess limitations in scalability to high dimension observation vectors. While MOCDHMMs are able to model covariance among the observation vector variables, such modeling

of covariance limits MOCDHMM ability to scale to large input vectors. To see why, consider the principle of machine learning that the more free parameters are in a model, the more training data is required to meaningfully fit them [23]. As the output vector of a MOCDHMM grows, the number of terms in the MOCDHMM covariance matrices grows faster; specifically, an output vector of  $k$  variables requires a covariance matrix of  $k^2$  elements. This polynomial increase in free parameters causes an increase in the amount of training data required as the observation vectors grow. That said, scaling to high dimensional spaces in MOCDHMMs is manageable if researchers have *a priori* knowledge that covariance is likely to exist only among certain variables. For example, if researchers expect  $\text{Cov}[a, b] = 0$  they can set the corresponding entries in the covariance matrix to 0 when initializing the Segmental K Means algorithm. Initialization to zero in this algorithm constrains the value to remain zero throughout training [41].

### 5.3 Future work

This section details several opportunities for future work based upon this research. Some opportunities are direct extensions of the work presented in this thesis which are most likely to advance the knowledge of applying MOCDHMMs to classifying entities in Automated Visual Surveillance (AVS) by entity track alone. Other opportunities would advance peripheral goals or advance the knowledge of MOCDHMMs in general. We treat opportunities for future work in order from those directly advancing MOCDHMMs for AVS to those advancing MOCDHMM use in general.

Following is a bulleted list of opportunities for future work. One subsection for each bullet gives additional details of the future work we recommend.

- Improve countermeasures for biased classifiers
- Discover other failure modes of MOCDHMMs
- Dataset selection

- Improve distribution analysis for scoring Log-likelihood
- Automatically select MOCDHMM model via Akaike Information Criteria (AIC) or Bayesian Information Criteria (BIC)
- Modify Baum Welch Re-estimation procedure to apply to MOCDHMMs
- Generalize experiments to permit five-at-a-time classification

### 5.3.1 *Improve countermeasures for biased classifiers.*

Table 4.7 shows that scored evaluation for five states underperformed unscored evaluation for four states by a statistically significant margin. However, also note from comparing Table 4.9 to Table 4.10 that increasing the number of hidden states in the HMMs generally increased the log-likelihood separability of the classes, subject to the bias problem for the SAFE HMM. We conclude that while increasing number of hidden states is (to a point) helpful in class separation, such an increase in hidden states causes classifier bias. Scored evaluation (as we present it) does combat classifier bias in the five state case. However, scored classification causes problems if the diagonal values of the log-likelihood matrices are not the highest in each column. Note that  $\mathcal{L}_\mu(\mathcal{S}^{\text{COMBAT}}|\lambda_{\text{COMBAT}})$  is not highest in the COMBAT column in the mean log-likelihood matrix in Table 4.10.

Future work may devise a method to combat biased classifiers which does not fail even when log-likelihood matrix diagonal values are NOT the highest in their columns. That is, future fixes for biased classifiers should help even when an HMM  $\lambda_A$  trained by sequence set  $\mathcal{E}^A$  recognizes some sequence set  $\mathcal{E}^B$  better than sequence set  $\mathcal{E}^A$ .

### 5.3.2 *Discovering other failure modes of MOCDHMMs.*

Although the techniques presented in this thesis were effective in preventing errors in several cases, another failure mode caused mathematical failures in the pilot experiment. A priority for future work is to determine the cause of these errors.

The results of the blended pilot experiment using the modified MOCDHMM algorithm (Figure 4.3) reveal cases in which the classifier produces near-random performance even in low blend densities. We have determined that these cases of low classification accuracy are due to a numerical error.

However, we have not determined the cause of such numerical errors. We believe the MOCDHMM algorithm in [41] does not account for some mathematical complexities of MOCDHMM use in actual datasets. It is worthy to note that similar cases avoid the nan error (compare results of agent C versus agent I in Figure 4.3). Neighboring cases alternately produce near-100% accuracy and near-50% accuracy. That the algorithm works in some similar cases but not in others suggests that a numerical error exists in the cases of poor accuracy. The cause of the near-50% accuracy remains an open question for future research.

### **5.3.3 Dataset selection.**

The Fryer dataset described in Section 3.6.3 is used in this research in part because it was a difficult dataset to model. The differences in pedestrian behavior from class to class were subtle; the Fryer Dataset simulates a real-world application in that there is not much difference in the track of various entities to be classified. However, the Fryer dataset fails to demonstrate disparate behavior among the classes that the Visual Battlespace 2 simulator and the JACK artificial intelligence framework [17] make possible.

Consider the descriptions of the agents in Section 3.6.3. SAFE agents behave exactly like CARELESS agents except when encountering an enemy, at which time they behave as AWARE. However, only one enemy is in play during the simulation that generated the Fryer dataset. Having only one enemy thus provides less separation between SAFE and CARELESS and thus less classification accuracy. AWARE agents are different from CARELESS agents in that they use cover to move from source to goal; however, dataset designers minimized terrain and cover features. CARELESS agents react to being directly



fired upon (as opposed to merely *encountering* an enemy) but enemies in this simulation are constrained to hold fire. As a result, AWARE agents and CARELESS agents often perform just like one another, which provides less separability. Therefore, while the Fryer dataset satisfied our requirement for a hard dataset to separate, it may have provided less separability than would be available in a real-world AVS problem.

To continue meaningful research into classification of agents by use of tracks, researchers should find better datasets. A dataset would be better to the degree that the separation in its classes relies on features most likely to occur in a real world intelligence analysis problem. An ideal dataset also have the following characteristics to permit classification based upon track:

- Tracked: The dataset will have logs of actual entity position  $(x, y)$  at a consistent sampling frequency
- Labeled: The dataset will contain an accepted ground truth as to which entities in it belong to which class
- High aspect or geospatially corrected: Much publicly available video data is taken from closed circuit TV cameras mounted low to the ground. In such a setup, targets moving away from the camera become smaller and closer to the horizon rather than registering a significant change in  $y$  position. Thus, even if tracked, the dataset might contain  $(x, y)$  coordinates that do not correspond to actual entity movement. Instead, the dataset should contain video taken from an aircraft or from a similarly high aspect. Alternatively, a dataset should be preprocessed to make geospatial corrections to low-aspect video

#### ***5.3.4 Improved distribution analysis for scoring log-likelihood.***

The scored HMM log-likelihood method given in Section 3.5.3 is a method that mirrors Z-score evaluation for Gaussian data. By dividing a log-likelihood's distance from

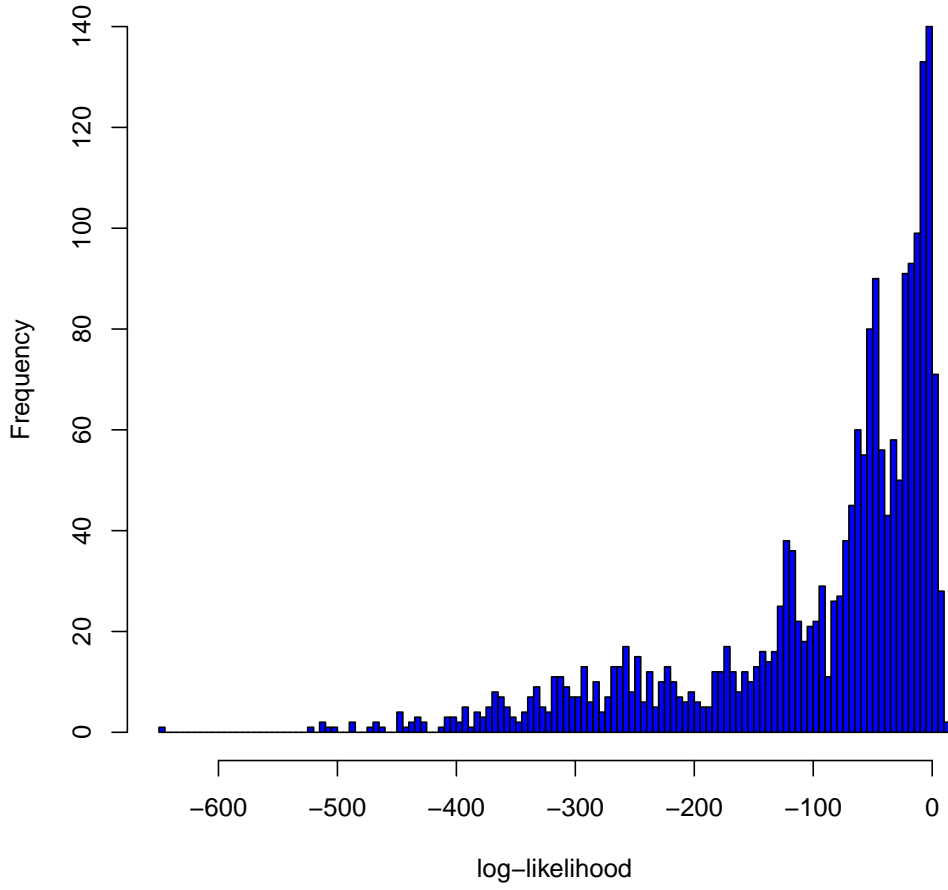


Figure 5.1: Histogram of distribution of log-likelihoods  $\mathcal{L}(\mathcal{S}^{\text{AWARE}}|\lambda_{\text{AWARE}})$  for AWARE sequences  $\mathcal{S}^{\text{AWARE}}$  given trained 5-state AWARE HMM  $\lambda_{\text{AWARE}}$ . Note that the distribution is not Gaussian.

the mean by the standard deviation, it implies the number of standard deviations from center is a good measure for how much of the distribution a particular log-likelihood is greater than. However, the distribution of log-likelihoods for a training sequence set are not necessarily Gaussian. To show an actual such distribution, we introduce Figure 5.1. Figure 5.1 gives a histogram of one set of log-likelihoods  $\mathcal{L}(S^{\text{AWARE}}|\lambda_{\text{AWARE}})$  for AWARE sequences  $S^{\text{AWARE}}$  given trained 5-state AWARE HMM  $\lambda_{\text{AWARE}}$ . The log-likelihoods of Figure 5.1 are neither Gaussian nor symmetric.

A more precise measure of where a given log-likelihood falls within the distribution may increase scored classifier accuracy. One possible approach would be to fit a distribution  $f(\mathcal{L}(S|\lambda))$  to the one given in Figure 5.1 and use Cumulative Density Function  $F(\mathcal{L}(S|\lambda))$  to determine what area of  $f(\mathcal{L}(S|\lambda))$  falls below the log-likelihood  $\mathcal{L}(S|\lambda)$  for each  $S$  to be classified. This may be a more appropriate score on which to classify  $S$  than was  $z(S|\lambda)$  described in Section 3.5.3. Such an implementation is an area for future research.

### ***5.3.5 Automatic model selection based on AIC or BIC.***

Experimentation alone is used in our research to determine the required number of hidden states in our HMMs. However, automated means exist to determine from training data the ideal number of hidden states. Both the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) are used to calculate the ideal number of model free parameters supported by the training data. Various research has applied this technique, such as that in [52]. However, our literature review has not found the application of AIC or BIC to tracking-based overhead Automated Visual Surveillance (AVS); such an application is an area for future work.

That this research required algorithm modification to generalize from simple HMM to MOCDHMM suggests AIC and BIC will require similar modification. It is important to

note that applying AIC and BIC to MOCDHMMs may not be a straightforward task even though AIC and BIC solutions for simpler HMMs already exist.

### ***5.3.6 Implement required mathematical changes in the Baum Welch Re-estimation procedure.***

Our early work in this research uncovered a variety of mathematical errors when we attempted to train MOCDHMMs using Baum Welch (BW) Re-estimation, even though our BW implementations for discrete inputs worked correctly. Because we discovered reliable means for Segmental K Means Training before we addressed BW deficiencies, we focused our efforts on Segmental K Means training alone. Finding solutions to problems with the BW algorithm for MOCDHMMs is an area for future work.

### ***5.3.7 Generalize experiments to permit five-at-a-time classification.***

As discussed in Section 4.3.3.2, no experimental data exists to permit direct comparison between Mean Euclidean Distance (MED) classification and MOCDHMM classification. The reason for the lack of comparable data is twofold: 1) Fryer [15] (who studied MED classification) did not perform pairwise classification on what we term the Fryer dataset and 2) we do not perform five-at-a-time MOCDHMM classification on the Fryer dataset.

In order to permit meaningful comparison between MED classification and MOCDHMM classification, our MOCDHMM classifier should be applied to a five-at-a-time experiment. Such a comparison might lend insight into whether a technique sensitive to the time-series nature of the dataset (like MOCDHMMs) is significantly better than a technique which is NOT sensitive to the time-series nature of the dataset (like MED).

## **Appendix A: Permission to Publish Proprietary Information**



**Technology Service Corporation**  
*an employee-owned company*

55 Corporate Drive 3rd Floor, Trumbull, Connecticut 06611 ■ Phone: (203) 601-8300 ■ Fax: (203) 452-0260 ■ [www.tsc.com](http://www.tsc.com)

*Ref: TSC-CT102-1031*

October 24, 2011

Matt Ross  
Air Force Institute of Technology  
Graduate School of Engineering  
2950 Hobson Way  
WPAFB, OH 45433

Dear Matt:

TSC is pleased that you wish to cite our SBIR Phase I Final Report on GMTI Exploitation Modeling: Deriving Behavior and Characteristics from Data Sets (AFRL-RI-RS-TR-2008-75) March 2008, Contract No. FA8750-07-C-0133. I understand that you will not reveal our detailed processing technique, by only wish to cite the benefits to warfighters that we claim in this report. If that is the case, then you have our permission to cite this work and we wish you the best with your publication.

Sincerely,

Allan Corbeil  
CT Operations Manager

AC/mf

# BLACK RIVER SYSTEMS COMPANY

November 30, 2011

Major Matthew Ross  
2950 Hobson Way  
Bldg 641  
Wright Patterson AFB, OH 45433

SUBJECT: Permission to Cite "GMTI Forensics Analysis Tools" Technical Report

This letter provides Major Matthew Ross permission to cite the below report in public academic literature.

Mark Kozak, Bryce Roskamp and Dr. Peter Shea, "GMTI Forensics Analysis Tools Final Technical Report", AFRL Technical Report under SBIR Phase I, AF071-061, Government Contract: FA8750-07-C-0100, March 2008.

Permission is granted to cite this paper regarding military capabilities the techniques would enable, so long as technical details of the methods are not revealed.

Sincerely,



Michael A. Krumme  
Executive Vice President  
Black River Systems Company, Inc.

## Appendix B: Proofs

Proof that  $\hat{\alpha}_t(i) = \prod_{\tau=1}^T c_\tau \alpha_\tau(i)$ :

*Proof.* The proof proceeds by induction. For  $t = 1$  we have

$$\alpha_1(i) = \pi_i b_i(\mathbf{O}_1) \quad (\text{B.1})$$

$$\hat{\alpha}_1(i) = c_1 \pi_i b_i(\mathbf{O}_1) = c_1 \alpha_1(i) = \left[ \prod_{\tau=1}^1 c_\tau \right] \alpha_t(i) \quad (\text{B.2})$$

by equations 3.7 and 3.11 respectively. For  $t = 2$  (base step) we have

$$\hat{\alpha}_2(i) = c_2 \omega_2(i) \quad (\text{B.3})$$

$$= c_2 \sum_{j=1}^N \hat{\alpha}_1(j) a_{ji} b_i(\mathbf{O}_t) \quad (\text{B.4})$$

$$= c_2 \sum_{j=1}^N c_1 \alpha_1(j) a_{ji} b_i(\mathbf{O}_t) \quad (\text{B.5})$$

$$= c_1 c_2 \sum_{j=1}^N \alpha_1(j) a_{ji} b_i(\mathbf{O}_t) \quad (\text{B.6})$$

$$= \left[ \prod_{\tau=1}^2 c_\tau \right] \alpha_2(i) \quad (\text{B.7})$$

For  $t > 2$  (inductive step), we assume the conclusion is valid for  $t$  and show it true for  $t + 1$  as follows

$$\hat{\alpha}_{t+1}(i) = c_{t+1} \omega_{t+1}(i) \quad (\text{B.8})$$

$$= c_{t+1} \sum_{j=1}^N \hat{\alpha}_t(j) a_{ji} b_i(\mathbf{O}_t) \quad (\text{B.9})$$

$$= c_{t+1} \sum_{j=1}^N \left[ \prod_{\tau=1}^t c_\tau \right] \alpha_t(j) a_{ji} b_i(\mathbf{O}_t) \quad (\text{B.10})$$

$$= c_{t+1} \left[ \prod_{\tau=1}^t c_\tau \right] \sum_{j=1}^N \alpha_t(j) a_{ji} b_i(\mathbf{O}_t) \quad (\text{B.11})$$

$$= \left[ \prod_{\tau=1}^{t+1} c_\tau \right] \alpha_{t+1}(i) \quad (\text{B.12})$$



□

Proof that  $\sum_{i=1}^N \hat{\alpha}_t(i) = 1$  for all time  $t$ :

*Proof.*

$$\sum_{j=1}^N \hat{\alpha}_t(j) = \sum_{j=1}^N \frac{\omega_t(j)}{\sum_{k=1}^N \omega_t(k)} = \frac{\sum_{j=1}^N \omega_t(j)}{\sum_{k=1}^N \omega_t(k)} = 1 \quad (\text{B.13})$$

□

Algebraic manipulation to obtain  $\mathcal{L}(S|\lambda) = -\sum_{t=1}^T \ln c_t$  from  $\mathcal{L}(S|\lambda) = \ln \left[ \frac{1}{\prod_{t=1}^T c_t} \right]$ :

*Proof.*

$$\mathcal{L}(S|\lambda) = \ln \left[ \frac{1}{\prod_{t=1}^T c_t} \right] \quad (\text{B.14})$$

$$= -\ln \left[ \prod_{t=1}^T c_t \right] \quad (\text{B.15})$$

$$= -\sum_{t=1}^T \ln c_t \quad (\text{B.16})$$

□

## Appendix C: Background on Automated Surveillance

In this appendix, we detail three topics relevant to Automated Visual Surveillance (AVS) which are peripherally relevant to the research in this thesis: occlusion, behavior based decomposition (BBD), and standards of method comparison. We cite papers relevant to each topic. In some cases, we refer to material presented in Chapter 2 in order to show how the material of this chapter relates.

### C.1 Occlusion and occlusion reasoning

Object based decomposition (tracking) methods described in Section 2.1.3 are susceptible to occlusion. *Occlusion* is the tendency of entities to hide one another from the camera or to be blocked from view by an obstacle. Some researchers claim that occlusion is such a significant problem that object based decomposition is infeasible for some applications. Loy et al. [34], in a paper developing behavioral based methods, declare that tracking methods are infeasible for “crowded wide-area scenes”. However, Kozak et al. [29] and Corbeil [5] present results that support the utility of object based decomposition in their work on Ground Moving Target Indicator (GMTI) data. The Kozak et al. and Corbeil methods in fact perform better for crowded scenes than for sparse scenes because the methods take advantage of crowded conditions to provide more data for accurate normalcy models. Occlusion becomes less a problem since [29] and [5] treat the aggregate case. Because not all methods leverage the aggregate case, many see decreased performance from occlusion when applying object based decomposition. (Techniques in [29] and [5] are proprietary and developed under contract to the U.S. government; thus, we do not present their implementation details.) Two techniques for *occlusion reasoning*, the process of contending with occlusion, are surveyed here: the Kalman filter and the temporal texture template.

The first technique for occlusion reasoning uses a Kalman filter [6] for reducing noise in data. Kalman filters are a smoothing method used in signal analysis. Kalman filters reduce random variation in received signals by considering two values: 1) the actual received signal value and 2) a predicted signal value calculated by the filter. The selected value for processing is that with the lower uncertainty.

A second technique for occlusion reasoning, temporal texture templates, is more recently developed. Temporal texture templates [20] maintain a state matrix on each tracked object. The matrix contains information on the foreground region of the entity and the frequency with which the pixels around that entity appear in the foreground of that object. Foreground region and frequency of change influence the next state of the state matrix as it is updated by the tracker. The machine detects when two tracked entities occlude, anticipating they will soon separate. When separation occurs, the present foreground and frequency of change of the two entities are compared to previous values. The closer matching images are then mapped to the tracked entity again. Stauffer et. al. [45] apply temporal texture templates effectively. Nonetheless, the Stauffer et al. method still performs less effectively as the number of occlusions increases.

## **C.2 Behavior based decomposition methods**

In contrast to object based decomposition (tracking) methods described in Section 2.1.3, *behavior based decomposition* (BBD) methods do not analyze the behavior of the entities by tracking them individually. Rather, BBD analyzes the changes of the scene pixels in a region. BBD models behavior or reports anomalies in a particular region rather than reporting the acts of a particular tracked entity as anomalous.

BBD has inherent advantages and disadvantages. Researchers apply the method to avoid the occlusion problem discussed above [16, 34]. Because BBD does not rely on statistical analysis of the entities, it produces acceptable results even when those entities change shape (e.g. get closer to the camera) or disappear entirely. In addition, BBD remains

computable even when predicting the behavior of a large number of objects since not all objects are modeled independently [34]. To realize these advantages, however, designers must provide for implementation issues specific to BBD: segmenting the video images and feature extraction. We shall describe both implementation issues.

BBD method designers, by contrast to tracking method designers, must decide how to segment video sequences appropriately into regions. Treating the sequences as one region might cause a method to miss information; segmenting into too many smaller regions introduces computability issues [34].

Feature extraction has more variables involved in BBD than in tracking. Almost all tracking methods extract the x-position and y-position of all tracked entities for each timestep. Instead, researchers applying BBD must decide what features in the larger video sequence are to be extracted if x-position and y-position are not. Many BBD methods [16, 34] are actually hybrid BBD/tracking methods which incorporate x-position and y-position if available. Other BBD features of interest may be: optical flow [34], dimension of entity bounding boxes [34], and variables based on Pixel Change History (PCH) [16]. In the Section C.2.1 we give more detail on feature extraction in BBD; in Section C.2.2 we give more detail on PCH.

### ***C.2.1 Feature Extraction.***

While it is important to note that both object based and behavior based decomposition perform some sort of feature extraction, the feature extraction methods vary more widely in BBD. Feature extraction, in this case, answers the question “how should relevant data be pulled from a video sequence if not by focusing on individual entities?” Two broad categories exist. Some methods use a compromise approach in which groups of moving pixels (“blobs”) are extracted. Some methods strictly analyze regions as a whole.

Decomposition into blobs is the most common method researchers use. Literature does not offer a rigorous, consistent definition for a blob. Our definition is “groups of pixels

that move together frequently enough that modeling them together provides the best chance to find anomalous events in an image”. Often, this leads to the drawing of a bounding shape (rectangle or ellipse) around the blob to approximate its size and shape. This blob is commonly reduced to a feature vector containing information such as centroid (x,y), size (width, height), filling ratio of foreground pixels within the bounding shape for the blob (occupancy), moments representing statistical likelihood of the blob being one coherent object, ratio of minor axis to major axis of the bounding shape, and general direction of motion (x,y) [34] [53]. There are inherent challenges to using this method. Each such variable will include noise if the BBD method cannot differentiate between two adjacent objects moving at a similar velocity [16]. Noise can cause a group of traffic on the highway to be identified as a single blob. Such a misidentification can raise false detections of anomalous behavior when the “object” suddenly splits [34]. Furthermore, analyzing so many blob variables can present computational issues since the pixels may be decomposed into far more blobs than are useful for machine analysis [16].

Pruteanu-Malinici et al. [39] detail a different feature extraction that analyzes video sequence regions as a whole. They claim superior performance using such a feature extraction. Pruteanu-Malinici et al. refer to the many-variable blob analysis above as *independent component analysis*. The Pruteanu-Malinici alternative, *invariant subspace analysis*, assumes that there is some feature in any video sequence that is not moving or changing. This *invariant subspace* is what Pruteanu-Malinici et al. avoid analyzing for anomalous behavior. The feature subspace of interest can then be represented as a set of orthogonal basis vectors,  $\mathbf{b}_t$  and a vector of pixel gray levels can be represented as  $\mathbf{x}$ . Then the invariant subspace is computed by:

$$s^{inv} = \sqrt{\sum_{t=1}^d \langle \mathbf{b}_t, \mathbf{x} \rangle^2}$$

Results in [39] validate the effectiveness of ISA. ISA performed from 12% to 50% better than independent component analysis in normal-events detection and abnormal-events detection [28].

### ***C.2.2 Reducing number of blobs considered: Pixel Change History.***

Various computability problems confront the BBD models. As discussed above, sometimes the feature extraction algorithms extract more blobs than are of interest, which slows computation. Pixel Change History (PCH) presents one method of reducing the number of irrelevant blobs extracted. One problem causing excessive blob generation is motion from the background that does not give insight into the behavior of the entities in the scene such as leaves waving in the wind. PCH addresses this problem by building a model to determine over how much time the motion takes place. PCH then applies a heuristic to separate events of significance. Short term changes are filtered from significance on the assumption that they are constant background motion such as leaves. Medium term changes are classified as new objects or major changes in already known objects. Long term changes are classified as new objects or previously known objects which have left the scene [53].

## **C.3 Comparing and Contrasting Approaches**

Though few operationally-fielded AVS solutions exist, a wide variety have been researched. The researched methods vary widely enough that comparison and contrast is difficult. This section attempts to provide a framework for comparing AVS methods in two ways. First, it discusses common metrics designed to evaluate performance. Second, it describes graphical mathematical models, which are used by almost all proposed solutions.

### ***C.3.1 Metrics.***

Because the discipline of AVS is fairly new, the measures of success for experiments in it are not standardized. Literature agrees that it is good to detect an anomaly and bad to detect an anomaly where none exists. From basic statistics, the former is termed *true*

*positive* and the latter *false positive* [30]. Notions of true positive and false negative have led to the only widely used standards of success—the receiver operating characteristic (ROC) curve (see figure C.1) and the confusion matrix.

The ROC curve is a plot of true positive rate against false positive rate for a particular solution. The higher the area under the ROC, the better the solution. However, ROC curves require that solutions can be calibrated to be more or less sensitive (to vary the false positive rate). A different measure of success, the confusion matrix, gives false positive, true positive, false negative, and true negative rates [7].

Unfortunately, characterizing AVS method performance by true positive and false positive alone ignores at least two points: 1) timeliness of alerts and 2) urgency of anomalous events detected [7]. First, characterization by true positive and false positive alone does not require the classifier to be timely in its output. For some applications, an abnormal event that is not detected in a timely manner is as bad as an anomalous event that is not detected at all. For example, a method that detects the placement of a bomb on a subway station does no good after the bomb has gone off. Second, characterization by true positive and false positive alone does not account for the fact that some abnormal events may be more important than others [7]. For application in homeland security, a method that detects the abnormal event of “littering” is not as effective as a method that detects the abnormal event of “placing bomb”.

The CREDS [55] (Challenge for Real-Time Events Detection Solutions) method evaluates anomaly detection for both timeliness and urgency. The CREDS challenge publicly issued video clips with anomalous events and called for solutions to classify them. (In one case, video clips involved security cameras on a subway station.) Solutions received a point value based upon their ability to detect anomalies in the video clips. Three failure modes caused point deduction: *anticipated detection* in which the classifier reports an anomaly before it happens, *delayed detection* in which the classifier is late to

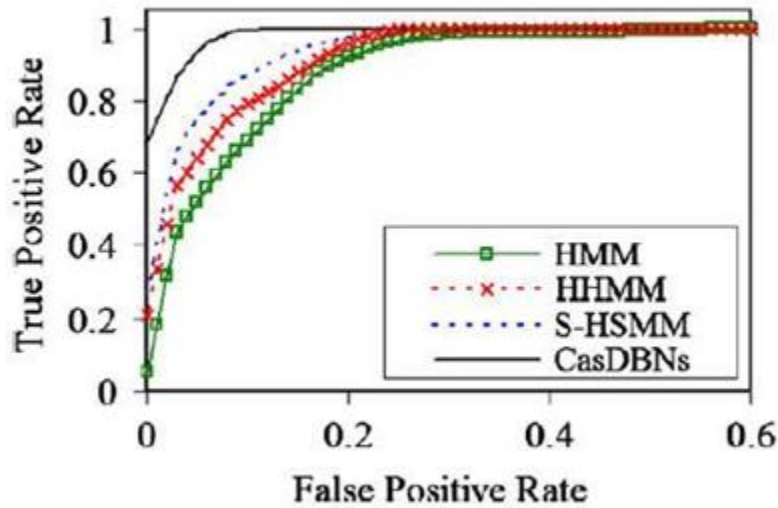


Figure C.1: Sample ROC Curve

detect, and *false negative* in which the classifier misses the anomaly entirely. Positive point awards for detection are proportional to how soon after the start of an event the detection occurred. Moreover, test designers designated certain events more critical than others, with correspondingly higher rewards and penalties for detect and miss. The CREDS evaluation method holds promise; no other methods we encountered in literature test both timeliness and urgency.

However, CREDS does have drawbacks. First, CREDS places a premium on the detection of events within seconds of their occurring. Such a premium is applicable when trying to save passengers who may have fallen on the tracks but not when detecting trouble in the next few hours. Second, CREDS evaluates only detection of events and does not demand a predictive element.



### *C.3.2 Graphical structures.*

The graphical structure is the part of an AVS algorithm that accepts whatever vectors feature extraction collects; the graphical structure produces the model or prediction. A variety of graphical structures exist and saw considerable evolution in AVS application from 1990 to 2010. In this section, we survey applications of graphical structures other than HMMs to AVS.

Gaussian Mixture Models (GMM) were among the first mathematical graphical structure used (late 1990s and early 2000s). Today, they still provide a benchmark against which other methods are tested. In 2000, Stauffer et al. [45] perform one of the latest experiments in AVS using GMMs without any other graphical models.

Certain modifications to Hidden Markov Models (HMM) produce new graphical structures. Gong [16] introduces the Multi-Linked HMM. Both Loy [33] and Oliver et al. [37] discuss the Coupled HMM (CHMM). Pruteanu-Malinici [39] makes a change to the HMM model to allow a theoretical infinite number of hidden states. Dynamic Bayesian Networks (DBN) represent a further evolution to the HMM paradigm. DBNs offer promise in very recent literature since their classification accuracy is on par with HMMs multiple times more complex [34, 53]. Simplicity and computability enabled by DBNs may be a key to analyzing wide area scenes and providing alerts in real time.

Statistical methods to determine what the best graphical model is for a given set of data have arisen in recent AVS literature. These techniques specify the ideal model complexity for a given set of data, and are not limited to any particular graphical structure. For example, Xiang uses Bayesian Information Criterion (BICr) and Completed Likelihood Akaike's Information Criterion (CL-AIC) to estimate the ideal model and complexity (e.g. number of states in an HMM) required [52].

## **C.4 Conclusion**

This appendix has surveyed several topics relevant to AVS. Material covered in this section gives context to the research of this thesis.

## Bibliography

- [1] AFOSI. “U.S. Air Force Eagle Eyes Website.” <http://www.osi.andrews.af.mil/eagleeyes/index.html>.
- [2] Ben-Israel, Adi and Thomas N. E. Grenville. *Generalized Inverses*. Springer-Verlag, 2003.
- [3] Brand, M., et al. “Coupled hidden Markov models for complex action recognition.” *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. 994–999. 1997.
- [4] Chen, J.L. and A. Kundu. “Unsupervised texture segmentation using multichannel decomposition and hidden Markov models,” *Image Processing, IEEE Transactions on*, 4(5):603–619 (1995).
- [5] Corbeil, Allan F., et al. *Ground Moving Target Indicator (GMTI) Exploitation Modeling: Deriving Behavior and Characteristics From Data Sets*. Technical Report AFRL-RI-RS-TR-2008-75, Air Force Research Laboratories, 2008.
- [6] Dee, H. M. and D. C. Hogg. “On the feasibility of using a cognitive model to filter surveillance data.” *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*. 34–39. 2005. ID: 1.
- [7] Dee, Hannah M. and Sergio A. Velastin. “How close are we to solving the problem of automated visual surveillance?,” *Machine Vision & Applications*, 19(5):329–343 (12 2008).
- [8] Dr. Gilbert Kuperman. Principal Mathematician, 711 HPW/RHXB, Wright Patterson AFB OH. Personal Interview. 19 May 11.
- [9] Duong, T. V., et al. “Activity recognition and abnormality detection with the switching hidden semi-Markov model.” *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. 838–845 vol. 1. 2005. ID: 1.
- [10] Easton, Morton L. *Multivariate Statistics: a Vector Space Approach*. John Wiley and Sons, 1983.
- [11] Fawcett, T. “An introduction to ROC analysis,” *Pattern recognition letters*, 27(8):861–874 (2006).
- [12] Fielding, Kenneth H. and Dennis W. Ruck. “Recognition of moving light displays using hidden Markov models,” *Pattern Recognition*, 28(9):1415 – 1421 (1995).

- [13] Forchhammer, S. and T.S. Rasmussen. "Adaptive partially hidden Markov models with application to bilevel image coding," *Image Processing, IEEE Transactions on*, 8(11):1516–1526 (1999).
- [14] Francois, Jean-Marc., "JAHMM: An Implementation of Hidden Markov Models in Java," 2006.
- [15] Fryer, Jr, Bryon K. *Virtual Battlespace Behavior Generation Through Class Imitation*. MS thesis, Air Force Institute of Technology, March 2011. Accession Number: ADA540365 Distribution Code: 01 - APPROVED FOR PUBLIC RELEASE Report Classification: U - Unclassified.
- [16] Gong, Shaogang. "Toward Behavior-Recognition-Based Video Surveillance," *Proceedings of the SPIE*, 5616:1 (2004).
- [17] Group, AOS. *JACK: An Agent Infrastructure for Providing the Decision-Making Capability Required for Autonomous Systems*. Technical Report, AOS Decision-Making Software, 2012.
- [18] Gupta, V., et al. "Integration of acoustic information in a large vocabulary word recognizer." *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.12*. 697–700. 1987.
- [19] Ham, Young Kug and Rae-Hong Park. "3D object recognition in range images using hidden markov models and neural networks," *Pattern Recognition*, 32(5):729 – 742 (1999).
- [20] Haritaoglu, I., et al. "W4: Who? When? Where? What? A real time system for detecting and tracking people." *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*. 222–227. 1998. ID: 1.
- [21] Huang, C.L., et al. "Gesture recognition using the multi-PDM method and hidden Markov model," *Image and Vision Computing*, 18(11):865–879 (2000).
- [22] Huang, X. D., et al. "Large-vocabulary speaker-independent continuous speech recognition with semi-continuous hidden Markov models." *Proceedings of the workshop on Speech and Natural Language*. HLT '89. 276–279. Stroudsburg, PA, USA: Association for Computational Linguistics, 1989.
- [23] Huang, X.D. and M.A. Jack. "Semi-continuous hidden Markov models for speech signals," *Computer Speech & Language*, 3(3):239 – 251 (1989).
- [24] Joint Chiefs of Staff. *Joint Publication 2-0*, 2007.
- [25] Juang, B.-H. and L.R. Rabiner. "The segmental K-means algorithm for estimating parameters of hidden Markov models," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(9):1639 –1641 (sep 1990).

- [26] Jung, SOH, et al. "Recognition of Alphabetical Hand Gestures Using Hidden Markov Model," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 82(7):1358–1366 (1999).
- [27] Kahneman, D and A. Tversky. "Subjective probability: A judgment of representativeness," *Cognitive Psychology*, 3(3):430–454 (1972).
- [28] Kohonen, Teuvo. "Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map," *Biological cybernetics*, 75(4):281 (10 1996). M3: Article.
- [29] Kozak, Mark P., et al. *GMTI Forensics Analysis Tools*. Technical Report, Black River Systems Company Inc. Utica, NY, 2008.
- [30] Kreyszig, Edwin. *One-Sided and Two-Sided Alternatives* (9th Edition), 1060. Advanced Engineering Mechanics, Hoboken, NJ: John Wiley and Sons, 2006.
- [31] Lee, C.-H., et al. "A study on speaker adaptation of the parameters of continuous density hidden Markov models," *Signal Processing, IEEE Transactions on*, 39(4):806–814 (apr 1991).
- [32] Lee, K.F. "On large-vocabulary speaker-independent continuous speech recognition," *Speech communication*, 7(4):375–379 (1988).
- [33] Loy, Chen Change, et al. "Time-Delayed Correlation Analysis for Multi-Camera Activity Understanding," *International Journal of Computer Vision*, 90(1):106–129 (10 2010). M3: Article.
- [34] Loy, Chen Change, et al. "Detecting and discriminating behavioural anomalies," *Pattern Recognition*, 44(1):117–132 (1 2011).
- [35] Morris, R. J. and D. C. Hogg. "Statistical Models of Object Interaction," *International Journal of Computer Vision*, 37(2):209–215 (2000). 10.1023/A:1008159822101.
- [36] Norris, C and G Armstrong. *The maximum surveillance society*. Berg, 1999.
- [37] Oliver, N. M., et al. "A Bayesian computer vision system for modeling human interactions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):831–843 (2000). ID: 1; additional tuning or training.
- [38] Povlow, B.R. and S.M. Dunn. "Texture classification using noncausal hidden Markov models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(10):1010–1014 (1995).
- [39] Pruteanu-Malinici, I. and L. Carin. "Infinite Hidden Markov Models for Unusual-Event Detection in Video," *Image Processing, IEEE Transactions on*, 17(5):811–822 (2008). ID: 1.

- [40] Rabiner, L. and B. Juang. "An introduction to hidden Markov models," *ASSP Magazine, IEEE*, 3(1):4–16 (1986).
- [41] Rabiner, L.R. "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, 77(2):257–286 (1989).
- [42] Rabiner, L.R., et al. "High performance connected digit recognition using hidden Markov models," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(8):1214–1225 (1989).
- [43] Rimey, R.D. and C.M. Brown. "Controlling eye movements with hidden Markov models," *International Journal of Computer Vision*, 7(1):47–65 (1991).
- [44] Samaria, Ferdinando and Steve Young. "HMM-based architecture for face identification," *Image and Vision Computing*, 12(8):537 – 543 (1994).
- [45] Stauffer, C. and W. E. L. Grimson. "Learning patterns of activity using real-time tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):747–757 (2000). ID: 1.
- [46] Tokuda, K., et al. "Multi-space probability distribution HMM," *IEICE Transactions on Information and Systems E series D*, 85(3):455–464 (2002).
- [47] Vogler, C. and D. Metaxas. "A framework for recognizing the simultaneous aspects of american sign language," *Computer Vision and Image Understanding*, 81(3):358–384 (2001).
- [48] Vstovsky, G.V. and A.V. Vstovskaya. "A class of hidden Markov models for image processing," *Pattern Recognition Letters*, 14(5):391 – 396 (1993).
- [49] Wallace, E. and C. Diffley. *CCTV control room ergonomics*. Technical Report Technical Report 14/98, Police Scientific Development Branch (PSDB), UK Home Office, 1998.
- [50] Wilson, A.D. and A.F. Bobick. "Parametric hidden markov models for gesture recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(9):884–900 (1999).
- [51] X.D.Huang, et al. *Hidden Markov Model for Speech Recognition*. Edmgurgh Univ. Press, 1990.
- [52] Xiang, Tao and Shaogang Gong. "Model Selection for Unsupervised Learning of Visual Context," *International Journal of Computer Vision*, 69(2):181–201 (08/15 2006). M3: Article.
- [53] Xiang, Tao and Shaogang Gong. "Video Behavior Profiling for Anomaly Detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(5):893–908 (2008). ID: 1.

- [54] Yamato, J., et al. “Recognizing human action in time-sequential images using hidden Markov model.” *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR’92., 1992 IEEE Computer Society Conference on.* 379–385. 1992.
- [55] Ziliani, F., et al. “Performance evaluation of event detection solutions: the CREDS experience.” *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on.* 201–206. 2005. ID: 1.

REPORT DOCUMENTATION PAGE					Form Approved OMB No. 0704-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>						
1. REPORT DATE (DD-MM-YYYY)		2. REPORT TYPE		3. DATES COVERED (From — To)		
14-06-2012		Master's Thesis		Oct 2010-June 2012		
4. TITLE AND SUBTITLE  Multi-Observation Continuous Density Hidden Markov Models for Anomaly Detection in Full Motion Video				5a. CONTRACT NUMBER		
				5b. GRANT NUMBER		
				5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S)  Ross, Matthew P., Major, USAF				5d. PROJECT NUMBER  12-312		
				5e. TASK NUMBER		
				5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB, OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/GCS/ENG/12-07		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Lt Col Lt Col Joshua Corner Program Manager - Anticipate and Influence Behavior Division Joshua.Corner@wpafb.af.mil—937-904-9548 Air Force Research Labs, 711th Human Performance Wing Bldg 248 2255 H Street WPAFB, OH 45433-7022				10. SPONSOR/MONITOR'S ACRONYM(S)  AFRL/711 HPW/RHX		
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED						
13. SUPPLEMENTARY NOTES This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.						
14. ABSTRACT An increase in sensors on the battlefield produces an abundance of collected data that overwhelms the processing capability of the DoD. Automated Visual Surveillance (AVS) seeks to use machines to better exploit increased sensor data, such as by highlighting anomalies. In this thesis, we apply AVS to overhead Full Motion Video (FMV). We seek to automate the classification of soldiers in a simulated combat scenario into their agent types. To this end, we use Multi-Dimensional Continuous Density Hidden Markov Models (MOCDHMMs), a form of HMM which models a training dataset more precisely than simple HMMs. MOCDHMMs are theoretically developed but thinly applied in literature. We discover and correct three errors which occur in HMM algorithms when applied to MOCDHMMs but not when applied to simple HMMs. We offer three fixes to the errors and show analytically why they work. To show the fixes effective, we conduct experiments on three datasets: two pilot experiment datasets and a simulated combat scenario dataset. The modified MOCDHMM algorithm gives statistically significant improvement over the standard MOCDHMM: 5% improvement in accuracy for the pilot datasets and 3% for the combat scenario dataset. In addition, results suggest that increasing the number of hidden states in an MOCDHMM classifier increases the separability of the classes but also increases classifier bias. Furthermore, we find that classification based on tracked position alone is possible and that MOCDHMM classifiers are highly resistant to noise in their training data.						
15. SUBJECT TERMS Artificial Intelligence; Automated Visual Surveillance; Anomaly Detection; Hidden Markov Models; Multivariate Gaussian Distributions; Feature Extraction						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON	
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lt Col Brett Borghetti (ENG)	
U	U	U	UU	120	19b. TELEPHONE NUMBER (include area code) (937) 255-3636 ext. 4612	